ML-GAT: Multi Label Node Classification Using Enhanced Graph Attention Networks

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELORS OF TECHNOLOGY
IN
COMPUTER ENGINEERING

Submitted By
Ashi Gupta (Roll number: 2k16/CO/065)
Prachi Garg (Roll number: 2k16/CO/220)

Under the guidance of
Dr. Rajni Jindal
Head of department
Department of Computer Science and Engineering

DEPT. OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY, DELHI
MAY 2020

ACKNOWLEDGEMENT

We would like to express our sincere gratitude towards our mentor and research supervisor Dr. Rajni Jindal, Professor and Head, Department of Computer Science and Engineering, DTU. We are grateful for her key insights, invaluable suggestions and constant encouragement which was instrumental in infusing self-assurance and helped us turn our ideas into sound shape and form. Without her support and dedicated involvement in every step throughout the process, this thesis could not have been accomplished.

We are grateful to the Department of Computer Science for giving us this research opportunity that has helped promote research at the undergraduate level. The crucial role of the staff of Computer & Software Engineering is also acknowledged with much appreciation, which helped us throughout the process of the development of this project by giving appropriate facilities and assistance. Lastly, we would like to thank our seniors and peers who helped us validate our ideas through important discussions that led to the successful completion of our thesis.

Ashi Gupta (2k16/CO/65)
Prachi Garg (2k16/CO/220)

ABSTRACT

Real-world graphs are ubiquitous data structures forming the backbone of a plethora of application domains, systems and phenomena ranging from bioinformatics and protein interaction to 3D modelling and learning for vision tasks. Graph neural networks are a promising class of message passing based neural network models that learn rich representations on graph-structured data in its raw form capturing complex entity relationships from the graph topological structure. While node classification using state-of-the-art graph neural networks is an active research direction, multi-label node classification on graphs is a relatively unexplored area. Since many real-world graph based problems require the assignment of more than 1 label to each node instance in the graph, we study here multi-label node classification using enhanced graph neural networks. We propose a novel architecture, Multi Label Graph attention Network (ML-GAT) that leverages the applicability of the attention based GAT to efficient inductive semi-supervised multi-label classification by augmenting complex inter-label and node-label dependencies implicit in the graph structure to the learning process. We compare our method with ML-GCN [2] and GAT [3] and GCN [1] baselines to examine the influence these losses have on the models and perform three empirical studies on the datasets to make a comparative analysis over the methods. After extensive optimization, we achieve significant performance increase on both the datasets from earlier benchmarks. We believe that this study will serve as a benchmark for future research in multi-label learning.

KEYWORDS

Geometric deep learning, Graph Attention Network, Graph Neural Network, Multi-label Classification

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

Social networks, recommender systems, 3D computer vision, molecular graph structures, telecommunication networks, and brain connectomes cannot be represented by a grid-like structure in a Euclidean domain. They are usually represented as arbitrarily structured graphs and fall under the category of non-Euclidean data. Prior to the advent of geometric deep learning and graph neural networks, there was no direct method to represent and capture graph topological information for training a model. It was done by using hand-engineered features representing structural information, which was time-consuming, costly and inflexible to dynamic learning, and adaptability to the learning process. Hence, it became necessary to have representation learning methods like deep learning that could directly work on graph-structured data.

Geometric deep learning allows algorithms to analyse information in its native form without any loss of inter-node relationships by generalizing deep neural networks to work on non-Euclidean data in the form graphs and manifolds. Central to any GNN is the concept of message passing in the form of neighborhood aggregation techniques, which can be used to leverage deep neural networks to learn relational knowledge on interacting entities in a large number of real-world applications and systems.

While most of the recent efforts in graph-based semi-supervised learning have been spent towards single-label classification, there are many real-world scenarios where each graph node could be simultaneously associated with multiple labels and complex label and node dependencies can be extracted from such data. These problems fall under the category of multi-label graph node classification.

Datasets which are defined using multiple labels for each data point are called multi-label datasets. Multi-label classification is the problem where each node instance can be assigned a subset of labels from the candidate label set. For example, in a social network, each user can belong to different social circles and in a collaboration network, each author could be working on multiple research domains.

Despite the literature concerning multi-label classification on graph-structured datasets, learning multi-label classification using GNNs which can capture the important correlation information from the graph structure is a relatively unexplored area and has been covered by very few papers. The paper [2] was the first to introduce a GCN based semi-supervised method for multi-label classification known as ML-GCN. It leverages a network made for single label classification (GCN) to learn efficient multi-label classification by incorporating inter-label and node-label co-dependencies. This paper serves as inspiration for our architecture where we elevate the capabilities and functionalities of the Graph Attention Network (GAT) by incorporating inter-label and node-label co-dependencies as done in their model ML-GCN to achieve a node representation learning form which performs well for multi-label node classification.

Multi-label classification is a semi-supervised learning problem as real-world graphs can be partially-labelled and we might not have access to all actual labels for all the nodes in the graph. In order to make our model flexible to unseen nodes (or unseen graphs altogether) and have wider application, we want an inductive semi-supervised graph neural network based multi-label classification technique. This is where the idea of adapting the ML-GCN model for a GAT baseline stemmed from as the GAT is inherently structured for inductive learning tasks via the use of self-attention mechanisms. This observation is supported by its inductive learning experiments on PPI dataset for which it achieves high performance on two completely unseen test graphs.

1.1.1 Contribution of our work

1. Firstly, we leverage graph attention networks for multi-label node classification and study the influence that the inclusion of label and node correlations has on the training process and learned models.
2. Secondly, we compare the relative importance of inter-label and node-label correlation with respect to ML-GCN and ML-GAT models.
3. Thirdly, we provide a comprehensive study that establishes state-of-the-art benchmarks for MLC using GNNs and standard datasets.
4. In our thesis, we highlight the motivation for selection of this topic, its innate relevance and importance in AI research and give a comprehensive analysis of each relevant research work done previously so as to learn from them and observe best practices. We have essentially tried to study models from the standpoint of what could work in MLC and performed extensive experimentation to observe changes, patterns and properties.

1.2 Preliminaries

1.2.1 Euclidean vs non-Euclidean geometry

Geometry is the realm of math which is constituted by elements like points, angles, lines, circles, squares, triangles and other shapes, as well as the mathematical properties and relationships between these elements.

Euclidean Data

The most commonly studied and known geometry is Euclidean geometry. Euclid was a 4th century Greek mathematician who invented Euclidean geometry by formalising and laying out the main principles of the field in his book 'The Elements'. It is the first to introduce axioms, theorems and proofs about squares, circles, acute angles, isosceles triangles, and other elements of Euclidean geometry. In the most basic terms, Euclidean geometry is the study of the geometry of flat surfaces. It is governed by three main properties that distinguish it from non-Euclidean geometry:

- One of the fundamental properties of Euclidean geometry is the Parallel postulate. The parallel postulate states that given a line and a point, there is only one other line that you can draw though the point that will be parallel to the original line. This is what we study when we study geometry in schools and is definitely true on a flat, two-dimensional surface, but this concept turns out to be too trivial and doesn't hold true in some other situations, including when the surface is curved.
- In Euclidean geometry, the interior angles of triangles always add up to 180 degrees
- Shortest distance between 2 points is a straight line joining the two points.

Hence, a Euclidean space can be defined as an n-dimensional space characterised by an infinite number of points, each point being represented by a coordinate in the n-dimensional space and distance between any two coordinates given by the distance formula.

Non-Euclidean Data

In the real world, not everything lives in a two-dimensional flat space and hence, not everything is bound by the laws of Euclidean geometry. Consider a simple example, a non-inflated balloon is a flat object, and is governed by Euclidean geometry. But inflate the same balloon, its surface is no longer flat and it can't be described by Euclidean geometry, and needs non-Euclidean geometry to define it mathematically. The most common non-Euclidean geometries are spherical geometry, elliptic geometry and hyperbolic geometry. The essential difference between Euclidean geometry and non-Euclidean geometries is the nature of parallel lines: non-Euclidean geometry doesn't follow the parallel postulate. Spherical geometry defines no such lines. In hyperbolic geometry at least two distinct lines that pass through the same point and are parallel to the given line.

Limitations of Euclidean geometry and traditional modelling approaches

Geometry is pervasive and fundamental in most real world data, structures and applications and a clear understanding is required to represent the data in the appropriate geometry so it can be deployed in training machine learning models to further artificial intelligence as the backbone of systems and to extend its applicability to the most uncommon data structures.

The core representational unit in the ML pipeline is the vector, and its multidimensional generalization, the tensor. In most machine learning and deep learning models, the natural choice for operable representation of data is to produce a matrix or vector of real numbers. But in this process, we have already imposed a choice of Euclidean geometry which holds true for flat surfaces only.

Consider the case of data structures having an inherent hierarchy or a tree like representation. Graphs and manifolds are ubiquitous and form the backbone of several real-world applications. These data types can't be represented by Euclidean geometry and traditional models (deep learning models specially) that only work with Euclidean spaces seriously limit their applicability and feasibility when it comes to learning on such data structures. This is because the space the points live in is limiting, and if we lift these models to the non-Euclidean geometry, we can break through these limits and unlock the full potential of models. When non-Euclidean data is represented and modelled by models built for Euclidean data, the operations require a large dimensionality and are a lot more complicated while non-Euclidean spaces can potentially perform these operations in a much more simple, straightforward and flexible manner, with fewer dimensions.

1.2.2 Inductive vs transductive learning

To fully understand graph neural network learning it is important to introduce the concept of induction and transduction in machine learning.

Inductive learning
In inductive learning, you analyse a set of data and try to make a generalisation in the form of a model or function that is able to predict the correct or close to correct labels on completely unseen inputs from the same or similar domain. You basically "induce" rules, characteristics and patterns from the data to learn a model that will work on new inputs. The idea is to get a good generalisation from the training data with the aim that it will generalise well on unseen samples. The learning can be supervised, semi-supervised or unsupervised. The output of the inductive learning process is in the form of a function that maps inputs to outputs. The domain of the inputs for this model learnt is not limited to the instances in the training data (or corresponding test data) that was made available to us for learning the model. It is constructed to work on any kind input instances in the future.

Transductive learning
Literally speaking, transduction is the transfer of something from entity A to entity B. Consider the problem where we are given a set P of labelled examples (x_i, y_i) where every x_i is the input vector, and y_i is the corresponding output label and a set P' of unlabelled instances x_i'. The target is to get the expected labels y_i'. for all instances in P'. In the inductive setting when we constructed a model that could label any future unseen inputs, we solved a problem that's more general than the one we needed to solve.

What transductive learning does is, it leverages the information contained in the training data P to make good, accurate predictions on the required unlabelled instances P'. It streamlines and focuses the process for a small set of target instances instead of trying to build a 'general' or universal model. Transduction, in the context of learning, refers to reasoning from specific observed training instances (like P), to specific observed (unlabelled) instances (like P'). It was introduced by Vladmir Vapnik, who explained it as:

"When solving a problem of interest, do not solve a more general problem as an intermediate step. Try to get the answer that you really need but not a more general one."

Basically, it avoids expanding the scope of the mapping to be learnt and by focusing it on the limited dataset, it gives better accuracy on that particular task. It's about doing only what is required and not compromising on the performance by trying to incorporate generalisations. Hence, the key difference between inductive and transductive learning is that induction refers to learning a general function that can be applied to any novel inputs, while transduction is more concerned with transferring some observed property onto a specific set of test inputs from the training data.

One of the main motivations behind our proposed model that uses graph attention networks for multi-label node classification is to develop an "inductive" semi-supervised technique that is not limited to the samples from the training data and is suitable for tasks like transfer learning on graphs when training data is hard to achieve and large scale inductive learning in tasks where there are missing node feature information and missing labels.

1.3 Graph fundamentals and traditional approaches

In this section, we formally introduce graphs and define graph-structured datasets. We then briefly explain the major approaches that use a graph embedding for deep learning. We then study the pitfalls of most of these techniques and explain how graph neural networks came into existence.

Graph theory defines a graph G as a set of V vertices (nodes) and E edges connecting the vertices in the graph. The nodes represent information holding entity objects in the graph while the edges represent relationships between the entities. The edges can be directed or undirected; weighted and non-weighted depending on the application case in point. Nodes and their relationships together represent the data by forming some expert knowledge or intuition about the problem. Nodes in graphs can be atoms in a molecule, users in a social network, cities in a transportation system, neurons in the brain, interacting objects in a dynamic physical system, points of the body used to model 3D manifolds or segmentation masks in images. All these applications involve training models on graph-structured data.

Prior to the advent of graph neural networks, several techniques were used to extend ML and DL models designed for Euclidean data to work on graphical data. Graph embedding is one of the key approaches that has been used in the past to transform nodes, edges, and their features into a Euclidean dimensional vector space. Fundamentally, it employs a class of graph pre-processing techniques with the sole aim to turn a graph into a computationally digestible format for the ML model whilst maximally preserving properties like graph structure and information in the transformed data. Embeddings can be performed at different granularity levels - node level, sub-graph level, or through strategies like graph walks. A few models worth mentioning are:

Deepwalk

Deepwalk [6] learns latent representations of vertices in a network by encoding graph information in the form of truncated random walks. These representations can easily be used in statistical ML models. The random walks are inspired from sequences of words (sentences) in language modelling. The steps of a random walk (a random graph traversal) are aggregated as instances in a matrix which can be input into a recurrent neural network. It is a transductive learning strategy that uses the skip-gram model to make predictions.

Node2Vec

Node2vec [27] was one of the first Deep Learning attempts to learn from graph-structured data. It works on the principle that If each node in a graph is transformed into an embedding like words in a sentence, a neural network can learn representations for each node. It features a walk bias variable α, which is parameterised by p and q. The parameter p prioritizes a breadth-first-search (BFS) algorithm, while the parameter q supports a depth-first-search (DFS) algorithm. The former is ideal for learning local neighbours, while the latter is better at learning global variables. Node2vec can switch to and from the two priorities depending on the task.

Graph2Vec

Graph2Vec [28] is a Node2vec variant that learns data-driven distributed representations of arbitrary sized graphs by learning to embed a graph’s sub-graphs. It's an unsupervised, task-agnostic, inductive approach that can be applied to many tasks like graph classification and clustering. It alleviates the limitations of handcrafted graph kernels which were previously used for such tasks but lack generalisation abilities.

Structural Deep Network embedding (SDNE)

The previous random walk based node embedding methods use shallow models which fail to capture high non-linearity in the model if the network structure is complex. Unlike such techniques, SDNE [29] learns using first order and second order proximity measures. Under first order proximity (pairwise similarity), two nodes sharing an edge are considered to be similar. Second order proximity is present if 2 nodes share many neighbouring nodes. By jointly optimizing these proximity measures in a semi-supervised deep model (made up of multiple layers of nonlinear functions), this method can preserve both the local and global network structure and is robust to sparse networks.

Large-scale Information Network Embedding (LINE)

LINE [30] minimises the difference between the input and embedding distributions using KL divergence. It explicitly defines 2 functions for first order and second order proximity. It defines two joint probability distributions for each pair of nodes in the graph and minimizes the KL divergence of the distributions. It fails to work when the task needs an understanding of node community structure.

Pitfalls of the traditional approaches

As can be observed by the approaches explained above, to extract structural information from graphs, traditional approaches mostly rely on summary graph statistics (*e.g.* degrees or clustering coefficients), kernel functions, or hand designed features to measure local neighborhood structures. They exhibit limitations because

1. These hand-engineered features are inflexible *i.e.* they cannot dynamically adapt during the learning process to heterogeneous graphs with different types and sizes
2. Designing these features could be an expensive process and time consuming.
3. They are transductive in nature mostly and can't be generalised to other applications domains where the graph is slightly different in nature. Also, addressing a different problem statement will be a problem (e.g. a model designed for node classification may not work for graph classification)
4. Many random walks approaches use shallow models that can't scale well to more complicated and large-scale graphs like the protein-protein interaction network dataset (PPI)

These shortcomings proposed the requirement of the highly-efficient representation learning methods on non-Euclidean data.

1.4 Graph Neural Network

Representation learning is a set of methods that allows a machine to be fed with raw input data and the model automatically discovers the representations needed for detection or classification on the data. Deep learning methods are typical examples of representation learning methods with multiple levels of representation, in the form of neural network layers that each transform the representation at one level into a representation at a higher, slightly more abstract level using simple but non-linear functions. It is to be noted that deep learning layers are not designed by human engineers, they are learned by the model from data using a general-purpose learning procedure that optimises a given objective function. If such models can be implemented for non-Euclidean data, it will resolve each one of the limitations pointed above.

While traditional deep learning research has focused on dealing with 1D, 2D, or 3D Euclidean-structured data such as acoustic signals, images, or videos, a large number of learning tasks require dealing with Modelling physics systems, learning molecular fingerprints, predicting protein interface, and classifying diseases which are in the form of graph data containing rich relation information among elements. **Extending neural network models to be able to efficiently and effectively learn on this kind of data is a very important direction for cutting edge deep learning research, but one that has received comparatively rather low levels of attention until very recently. This is the subject of our thesis wherein we explore a particular problem statement that is best solved by leveraging existing deep learning techniques using graph neural networks and demonstrate the significant advantages of using geometric learning for non-Euclidean spaces and its impact in artificial intelligence research.**

Geometric deep learning is a term for the unique and state-of-the-art emerging techniques attempting to generalize deep neural network models to non-Euclidean domains such as graphs and manifolds. Graph neural networks (GNNs) models capture the dependence of graphs via message passing between the nodes of graphs. Unlike standard neural networks, GNNs retain a state that can represent information from its neighbourhood with arbitrary depth.

CHAPTER 2

LITERATURE REVIEW

2.1 Related Works

In this section, we first describe important previous work that has led to the discovery and evolution of this field of geometric deep learning. The key graph neural networks have been listed along with the most relevant frameworks that address the issue of multi-label classification on graph-structured data using neural networks.

One of the first models to learn relational information between nodes of a network was DeepWalk [6]. DeepWalk uses random walks in the network (like structured sentences) to learn the relational structure of the dataset. Although DeepWalk gives good results for multi-label classification (MLC) on social networks, it is outperformed by PLANETOID [21] which is a semi supervised inductive learning technique that learns an embedding jointly trained for classification and graph context prediction. Due to the limitations of these methods, efforts were made towards GNNs.

2.1.1 Graph Neural Networks

The first neural network capable of directly processing graphs was Gori et al [8] which defined a graph neural network (GNN) as an extension of recursive neural networks. GNNs can be broadly classified into spectral based and non-spectral based approaches. Graph based spectral approaches are based on representing the graph in a spectral form defined in the Fourier domain. Bruna et al [9] used a CNN in a spectral representation which reduced the number of parameters resulting in faster forward propagation. It's shortcoming of large computational requirements because of Laplacian calculations is reduced by Defferrard et al [10] through Fast localized Spectral Filtering.

Further reductions in the computational requirements were made with the help of a model introduced by Kipf and Welling [1] called Graph Convolutional Network (GCN). In this model first order approximation of spectral representation of the graph is used. Non-spectral approaches enabled input of the architecture to be of arbitrary shape and size. This was introduced in Duvenaud [11] by using an individual weight matrix for each node degree. While [11] only allowed using a squared matrix for parameters, Atwood and Towsley [12] removed this limitation by introducing the diffusion-convolution operation for graphs. After this MoNet [13] was introduced, which generalized all the previous graph convolution techniques.

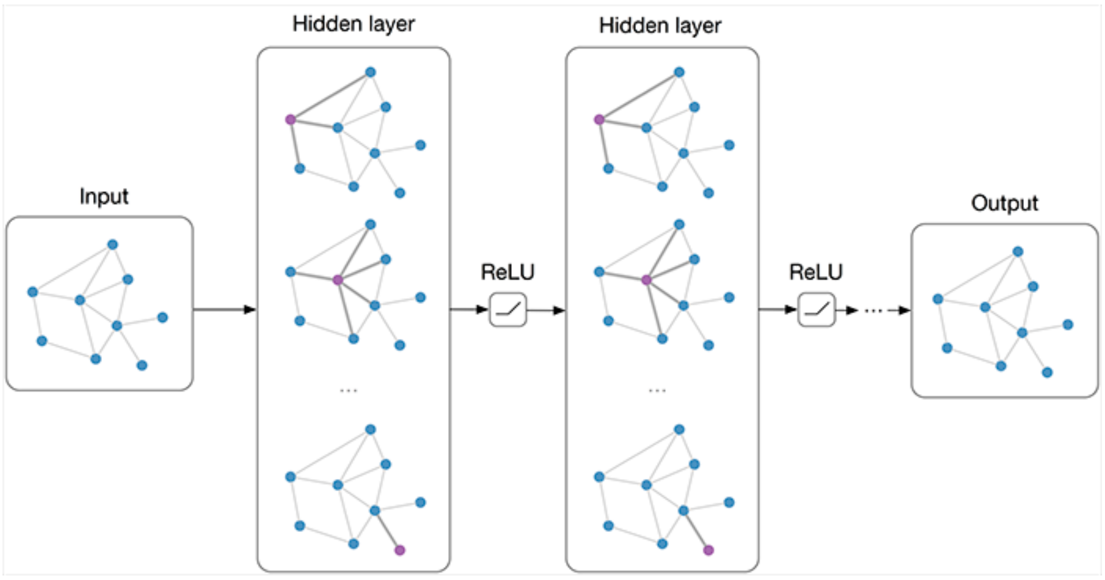

Figure 2.1 A two-layer Graph Convolutional Network. The pink dots denote the node for which neighbourhood aggregation is taking place.

GraphSAGE [14] was introduced as an inductive framework, which works on unseen data by performing sampling and aggregation operations like mean, LSTM, and pooling. To overcome the limitations of earlier spectral based approaches, Graph Attention Networks [3] introduced a multi-headed self-attention based neighborhood aggregation scheme which proved well suited for both transductive and inductive tasks.

The central operation in GNNs is the neural message aggregation and passing used in forward propagation to capture structural information efficiently. Most GNNs differ in the type of aggregation scheme used. Graph convolutional networks are broadly classified as spectral and spatial networks. Another important category is graph attention networks that use attention mechanisms in the neighbourhood aggregation step. Graph Spatial-Temporal Networks are used in time series prediction problems like STGCN [26] has been used for accurate traffic forecasts. Graph Auto-encoders combine the encoder-decoder pairs using graph representations on both sides. Graph generation can be central to many domains and Graph Generative Networks aim to generate all plausible structures from given data.

2.2.2 Multi-Label Classification on Graphs

The papers [16] and [17] showed that in multi-label relational datasets, label sets of related nodes and labels themselves are not independent. In [16], they capture single label and cross label dependencies among related instances and capture inter-instance and intra-instance label-label dependencies to enhance performance for multi-label collective classification. Akujuobi et al [15] is a reinforcement learning based multi-label classification model for attributed graphs that uses simultaneous collaborative graph walks by multiple label specific agents. Although training multiple binary classifiers for each label is a straightforward approach, it is not preferred because the complexity increases as the number of labels increases and has scalability issues.

The paper [20] models multi-label classification on a Siamese GCN network with the two branches of the network addressing inter-label and node-label interaction. The way it incorporates node-label and label-label interaction into the learning process is significantly different from [2] and the model we propose here. Another multi-label classification model for graph datasets is MAGNET [19]. This model is used for text classification and uses the dot product of GAT and biLSTM outputs to predict labels. One major differences between our model and MAGNET is that MAGNET is suitable for only text classification whereas our model covers a wider application domain.

2.2 Key Application Areas

2.2.1 Geometric Deep Learning On Graphs

Geometric deep learning encompasses learning on graphs and networks as well as 3D modelling and learning on manifolds. A few application areas include mining relation information in graphs like a railways network in a large metropolitan city; community prediction problems like Zachary's Karate Club; researchers have applied GCNs combined with reinforcement learning to the classic combinatorial optimisation problems like travelling salesman problem and predicting unknown molecular structures of complex organic molecules in chemistry.

In 3D modelling, GNNs can process kernels over the 3D point cloud. Performance on common image tasks can be significantly improved if the same problem is shifted to 3D instead of 2D images as images restrict the data to a single perspective angle. Many applications will perform better because their inherent structure requires a 3D model rather than a 2D projection. Following are a few examples to showcase the power of GNNs:

Interaction networks in physical systems

Daily phenomenon and processes in the real world can be modelled as entities and their interactions in a graph to aid reasoning and build predictive models on them. Object interaction in a complex physical system can be reasoned using Interaction networks. It can make inferences and predictions about complex system properties in domains such as collision dynamics. The system simulation is performed using relation and object centric reasonings by using deep neural networks on graphs. GNNs have been shown to predict the accurate trajectories of objects in the physical system thousands of time steps into the future.

Few shot learning on images

A standard deep learning problem, image classification assigns labels to images containing objects belonging to a given number of classes. Mostly, it has been performed using convolutional neural networks (CNNs) and CNNs have achieved state-of-the-art performance accuracy in a wide range of image classification tasks. However, zero shot learning and few shot learning are still evolving techniques wherein, the goal is to construct a classification model that can be used to predict labels that it has not seen previously (in case of ZSL) or seen very few (single shot and few shot learning) instances during training. We can model this task using GNNs where Knowledge graphs can provide the necessary information to guide the ZSL task. These graphs may be based on inter-image or inter-object similarities or incorporate semantic information of the category labels or captions of images. Graph neural networks can then be applied to structured data to aid a few shot learning image classification or object detection tasks.

Predicting side effects due to drug interaction

Decagon's graph convolutional neural network model [25] is a multi-relational link prediction approach for multimodal networks with a large number of edge types. Polypharmacy is prescribing a combination of multiple drugs to the same person to cure multiple problems. In this study conducted by researchers at Stanford University, Decagon architecture has been used to model pharmacology side effects computationally. Today, several people are hospitalised and succumb to negative side effects of combination drugs prescribed to them. Given the proliferation of pharmaceuticals, it is not possible to practically test each combination of drugs for interaction side effects.

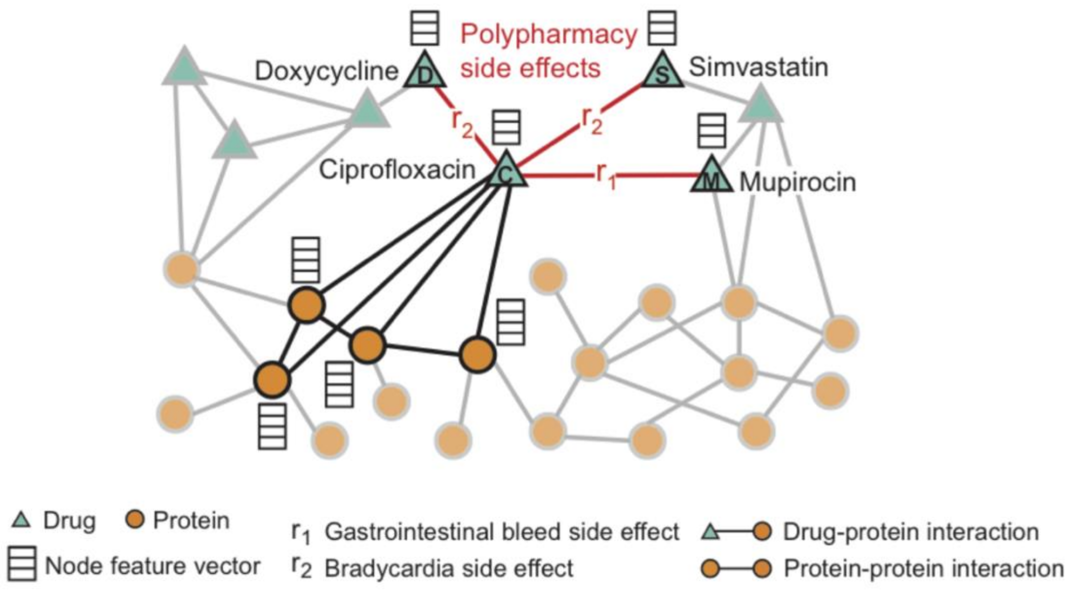
Polypharmacy side effects
Doxycycline
D
Simvastatin
S
r_2
r_2
Ciprofloxacin
C
r_1
M
Mupirocin
Drug
Protein
Node feature vector
r_1 Gastrointestinal bleed side effect
r_2 Bradycardia side effect
Drug-protein interaction
Protein-protein interaction

Figure 2.2 An illustration of sample graph of polypharmacy side effects derived from patient population data. In the graph, we can observe protein-protein interaction, drug-protein interaction and drug-drug interactions that encode 964 side effects (target labels). [24]

In practice, doctors rely on prior medical history of patients and prescribe medicine based on their knowledge of existing side effects. Polypharmacy side effects emerge as a result of drug-drug interactions, wherein activity of one drug may change, often unfavourably, if taken with another drug. Discovering polypharmacy side effects is a challenging problem which has been tackled by this research which constructs a multimodal graph of protein-protein interactions and drug-protein target interactions, and uses a GCN variant to predict any possible side effects.

Citation networks

Citation network can be of two forms. In the first form, research articles and their inter-paper citation can be modelled as a citation graph where the citation information can be used to improve the task of classifying the papers into research domain categories like 'computer vision' and 'deep learning'. The other form can be of an authorship network where authors from nodes in the graph and edges capture co-authorship information between the authors. These can be used to predict which subject areas the authors are publishing/researching in and authors closer to each other in the graph are more likely to be working on similar and related areas.

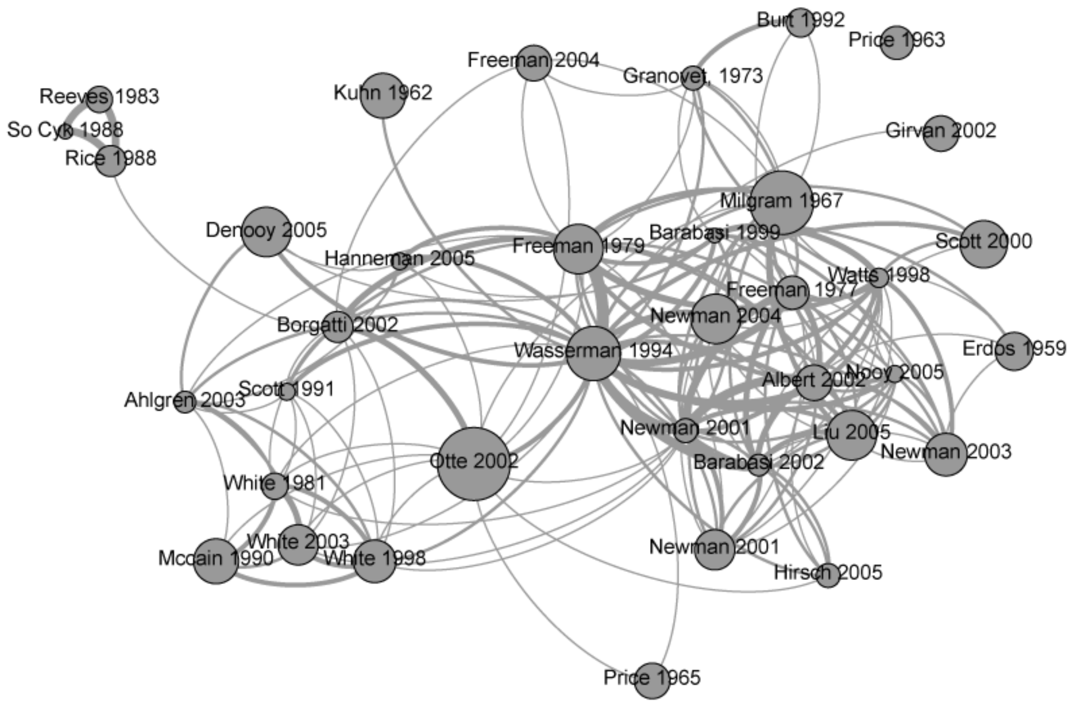

Figure 2.3 An example of a co-citation network for social network analysis, courtesy [https://eduinf.eu/2012/03/15/co-citation-analysis-of-the-topic-social-network-analysis/]

2.2.2 Multi-label node classification applications

In multi-label node classification, we can assign multiple labels to each instance. This is different from single label node classification where each instance can belong to a single class from the possible classes. To highlight the feasibility and advantages of using GNNs for multi-label node classification, we consider a few case studies where graph neural networks have showed promise for multi-label classification tasks:

Protein protein interaction networks

Protein protein interaction networks are commonly used to study amino acid interactions and GNNs can solve several problem statements on this dataset. As an example, the paper Protein Interface Prediction using Graph Convolutional Networks [24] tackles the challenging problem of prediction of interfaces between proteins. This problem has key applications in drug discovery and design and the authors have analysed using several graph convolutional operators on the dataset to get results that are better than the state-of-the-art SVM used previously.

Semi-supervised Multi-label image classification [23]

In problems like video annotation, a video clip can be annotated with multiple labels at the same time, such as 'person', 'people march', 'pedestrians'. This paper proposes graph based learning architecture that simultaneously explores the correlations among multiple labels and the label consistency over the graph to perform multi-label image classification and video annotation. Their model exceeds all previous performance benchmarks over the TRECVID 2006 corpus.

3D Facial Expression Recognition [22]

In facial expression classification, a person may express happiness and relaxation at the same time. Hence, it is a multi-label classification problem. Automatic facial behaviour analysis, including facial expression of emotion and facial action unit (AU) recognition can be regarded as pivotal to next-generation computing systems as it plays a key role in proactive user interfaces, learner-adaptive tutoring systems, personal well-being technologies, etc.

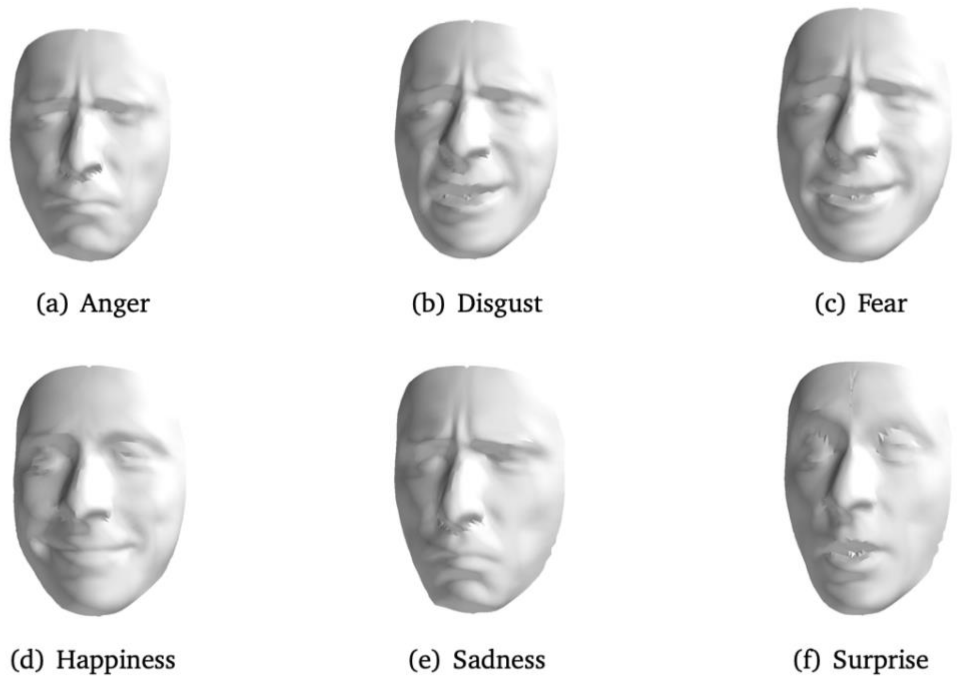

Figure 2.4 3-D facial expressions of a participant in the 4DFAB dataset [22]

The majority of prior work conducted in this area involves 2D images, despite the limitations due to illumination variations and inherent pose. In order to deal with such problems, 3D faces containing much more information than a flat image are increasingly used in expression analysis. With regard to 3D facial expression recognition, [22] proposes a deep residual B-Spline graph convolution network, which allows for end-to-end training and inference without using hand-crafted feature descriptors and significantly improves on image based approaches on the high-resolution 3D face dataset 4DFAB. This is an example of how geometric deep learning on 3D manifolds is revolutionising interactive AI and augmented reality today.

CHAPTER 3

PROPOSED WORK

3.1 Present models

This section describes the existing GNN frameworks which form the baselines of our proposed model

3.1.1 Graph attention network

In this section, we describe the building blocks of this neural graph processing framework that is a well-established network in the field of graph neural networks. At the core of the architecture is the graph attentional layer which uses an attention-based neighbourhood aggregation mechanism unlike spectral based approaches like GCN.

Graph Attention Network(Fig 3.1), comprising neural network architecture that operates on graph-structure data (geometric deep learning). Graph attention networks use the concept of masked self-attentional layers to take advantage of data inherent with relationships, connections and shared properties.

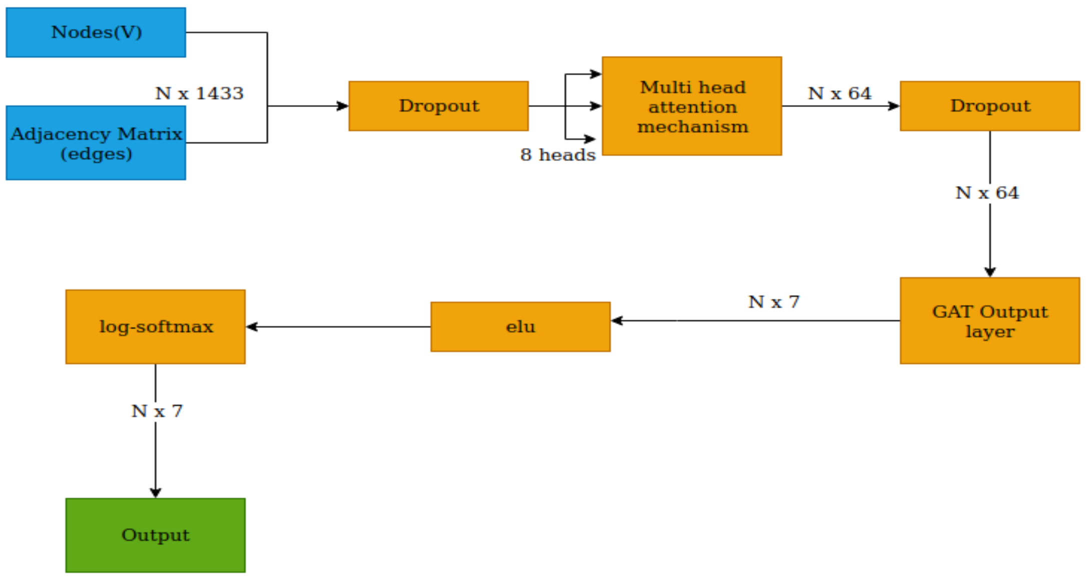

Figure 3.1 Structure of Graph Attention Network

Attention function mainly gives importance to the input states in which it has more contextual relations. Further in attention the contextual vector is built in such a way that it gives a global level of information of all the inputs i.e., relations are preserved and used for calculation at a global level. Attention mechanisms allow dealing with variable sized inputs, focusing on the most relevant parts of the input to make decisions. Attention mechanism used to compute a representation of a single sequence, it is commonly referred to as intra-attention or self-attention.

The attention architecture has the following interesting properties:-

1. The parallelizability of the operation across the nodes it efficient .
2. Arbitrary weights to the neighbours can also be specified so as to handle nodes with different degrees.
3. The model can handle inductive learning tasks easily.

Graph attention layer

The input to this layer is $,h = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_n\}, \vec{h}_i \in R^F$,where defines the total number of features available for each node and represents the total number of nodes the layer gives as output a new node feature representation in the form of $h' = \{\vec{h'_1}, \vec{h'_2}, \ldots, \vec{h'_N}\} \in R^{f'}$.

After applying a shared linear transformation parameterised by a weight matrix, $W \in R^{F' \times F}$, applied to each node, self-attention is performed on the nodes using shared attentional mechanism $a : R^{F'} \times R^{F'} \rightarrow R$ which computes attention coefficients :

$$e_{ij} = a\left(W\vec{h}_i, W\vec{h}_j \right) \tag{3.1}$$

For graphs with weighted edges, calculation of attention coefficients involves use of weights of the edges $\overrightarrow{E^w_{ij}}$ as follows :

$$e_{ij} = a\left(\overrightarrow{W_{h_i}}, \overrightarrow{W_{h_j}}, \overrightarrow{E^w_{ij}} \right) \tag{3.2}$$

Here , a is the alignment function (a feed forward jointly trained with the model) and e is the alignment score in this case attention coefficient which indicates the importance of node j's features to node i. Graph structure in injected into the mechanism by performing masked attention i.e., e_{ij} for nodes $j \in N_i$, where N_i is neighborhood (first order neighbours) of node i in the graph.

Attention coefficients are then normalized to make them easily comparable :

$$\alpha_{ij} = softmax_k(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in N} exp(e_{ik})} \tag{3.3}$$

In graph attention networks the attention mechanism is a single layer feedforward neural network parametrized by weight vector, and LeakyRelu nonlinearity is also applied.

$$\alpha_{ij} = \frac{exp\left(LeakyRelu\left(\vec{a}_T\left[\overrightarrow{W_{h_i}} \parallel \overrightarrow{W_{h_j}}\right]\right)\right)}{\sum_{k \in N_i} exp\left(LeakyRelu\left(\vec{a}_T\left[\overrightarrow{W_{h_i}} \parallel \overrightarrow{W_{h_j}}\right]\right)\right)} \tag{3.4}$$

$$\alpha_{ij} = \frac{exp\left(LeakyRelu\left(\vec{a}_T\left[\overrightarrow{W_{h_i}} \parallel \overrightarrow{W_{h_j}} \parallel \overrightarrow{E^w_{ij}}\right]\right)\right)}{\sum_{k \in N_i} exp\left(LeakyRelu\left(\vec{a}_T\left[\overrightarrow{W_{h_i}} \parallel \overrightarrow{W_{h_k}} \parallel \overrightarrow{E^w_{ik}}\right]\right)\right)} \tag{3.5}$$

The complete attention mechanism producing normalised attention coefficients is given by (3.4) for graphs with unweighted edges and (3.5) for graphs with weighted edges.

The obtained normalized attention coefficients are used to compute linear combinations of the features corresponding to the attention coefficients, this is used as the final output features for every node:

$$\vec{h}_i = \sigma\left(\sum_{j \in N_i} \alpha_{ij} W \vec{h}_j\right) \tag{3.6}$$

Regularisation

Multi-head self-attention is used in the Graph attention network to provide the model to capture various aspects of the graph structured data and improve its expressive ability. Essentially , the multi head attention is a combination of several self-attention layers stacked in parallel, with linear transformations of the same input.

Independent attention mechanisms execute the transformation(3.7), and then their respective features are concatenated:

$$\vec{h}'_i = \overset{K}{\underset{k=1}{\parallel}} \sigma\left(\sum_{j \in N_i} \left(\alpha^k_{ij} W^k \vec{h}_j\right)\right) \tag{3.7}$$

Here || represents concatenation and α^k_{ij} represent normalized attention coefficients.

While employing multi-head attention on the final layer of the network, averaging is employed instead of concatenation.

3.1.2 Multi-label Graph Convolution network

In this section, multi-label graph convolution network is described, the basic architecture of ML-GCN is used as inspiration for our project. Multi label graph convolution network uses graph convolution network to first embed graph topological information and node features.

Although GCN produces respectable embedding results, it simply reduces cross entropy loss between the last layer output and actual label representation, GCN alone is not capable of learning embedding suitable for multi label classification. Hence ML-GCN model focuses on developing on basic GCN architecture so as to enable multi-label classification.

ML-GCN focuses on two aspects, one of which is present in the form of dependencies between node instances and their respective labels because certain types of node features characterize particular subsets of labels. Henceforth, this dependency will be referred to as node-label correlation. The other type of dependency exists between nodes whose labels frequently occur together in the dataset. For example, in a collaboration network if the research area tags 'Domain adaptation' and 'Computer vision' frequently occur together, then it is likely that the authors publishing in these areas are closely situated in the lower dimensional feature space. Hence, capturing such label-label correlations can help us get richer feature representations from the model which are well suited for MLC.

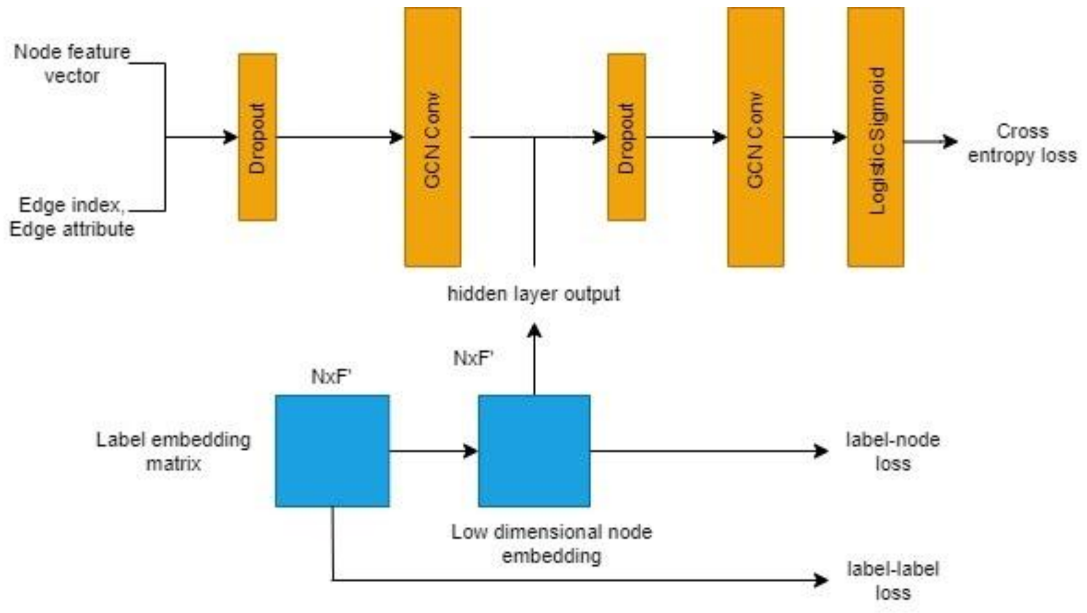

Figure 3.2 Network architecture for Multi-Label Graph Convolutional Network (ML-GCN)

Architecture of ML-GCN is shown in figure 3.2.ML-GCN achieves this by generating a uniform label-node co-embedding in a lower dimensional space using which the node-label and label-label correlation losses can be easily computed for use in model optimisation. This is done by introducing a label embedding matrix. The label embedding matrix is represented as $Z_Y \in \mathbb{R}^{N \times L}$, where L represents the number of labels.

This randomly generated label embedding matrix is used to calculate the label-node loss and the first embedding matrix generated from GCN along with the label embedding matrix is used to calculate the label-node loss. To maximize the probability of occurrence of labels for their respective nodes Skip-gram model is used. Skip-gram model is used to find context words using a target word. Skip-gram model on graph could be done by considering node of the graph as target word and labels of the node as context words for that node, hence forming a sentence of the node and its labels. Sentences formed using nodes along with labels are used to minimize label-node loss whereas sentences formed using label pairs are used to minimize label-label loss.

Let $Y_{x_i} = \{y_1, y_2, \ldots, y_c\}$, represent the label vector for node x_i and $\vec{h}'_i$ represent the feature vector output of second last layer of GCN for node x_i. Also, label vector of y_j is represented as z_{y_j} . We use node-label pairs ,$\{(x_i y_1), (x_i y_2), \ldots, (x_i y_c)\}$, for optimization of node label embedding by maximizing the following equation :

$$max \frac{1}{c} \sum_{y \in Y_{x_i}} \log P\left(z_{y_j} | \vec{h}_i \right) \tag{3.8}$$

To optimize label-label correlation label-label pairs , $\left\{ \left(y_1 y_2 \right), \left(y_1 y_3 \right), \ldots, \left(ycy_{c-1} \right) \right\}$, are used by maximizing the following equation:

$$max \frac{1}{c} \sum_{y_i, y_j \in Y_{x_i}, i \neq j} \log P\left(z_{y_j} | z_{y_i} \right) \tag{3.9}$$

For node-label loss, all possible pairs of a node and its labels are constructed just like word pairs in the skip-gram model and a co-occurrence probability is calculated. Similarly, intra-label pairs are made for computing label-label loss co-occurrence probability. The summation over all the node-label and label-label pairs is a costly step and the time complexity would increase with the number of labels. To alleviate this, a negative sampling technique is used. This negative sampling is used to calculate $P\left(z_{y_j} | \vec{h}_i \right)$ and $P\left(z_{y_j} | z_{y_i} \right)$ which needs to be calculated over all the nodes , negative sampling reduces the running time. This results in calculation of label-label and label-node loss. Which are further used to optimize the model for MLC.

3.2 Proposed Framework

Our proposed Multi-Label Graph Attention Network (ML-GAT) model is constructed by using the concepts of the ML-GCN architecture [2] on the Graph Attention Network baseline to elevate it for use in a multi-label node classification problem. We reiterate that MLC is a complex process where many kinds of relationships need to be learnt from the graph in order to learn a truly representative and robust model. These relationships include label-label and node-label correlations. This node-label and label-label correlation information is not explicitly available in the graph structure and hence cannot be captured by the basic GAT model.

3.2.1 Intuition

Intuitively, this model should perform well on the graphical multilabel node classification problem because of the ability of GAT to perform inductive learning tasks. In order to make our model flexible to unseen nodes (or unseen graphs altogether) and have wider application, we want an inductive semi-supervised graph neural network based multi-label classification technique. This is where the idea of adapting the ML-GCN model for a GAT baseline stemmed from as the GAT is inherently structured for inductive learning tasks via the use of self-attention mechanisms. This observation is supported by its inductive learning experiments on PPI dataset for which it achieves high performance on two completely unseen test graphs..

3.2.2 Training

The input to our architecture is a set of node features $h=\{\vec{h_1},\vec{h_2},..,\vec{h_N}\}\in R^f$ and a graph adjacency matrix that captures edge information(citations). Like in the existing GAT framework, the input is passed through a multi-head attention layer (8 multi-head units) called a graph attention layer. The layer output is a concatenation matrix of node features $h' = \{\vec{h'_1}, \vec{h'_2}, \ldots, \vec{h'_N}\} \in R^{f'}$ formed after performing the self-attentional neighbourhood aggregation mechanism.

Taking inspiration from ML-GCN , our model focuses on generating uniform label-node co-embedding for optimization of label-label and node-label loss .

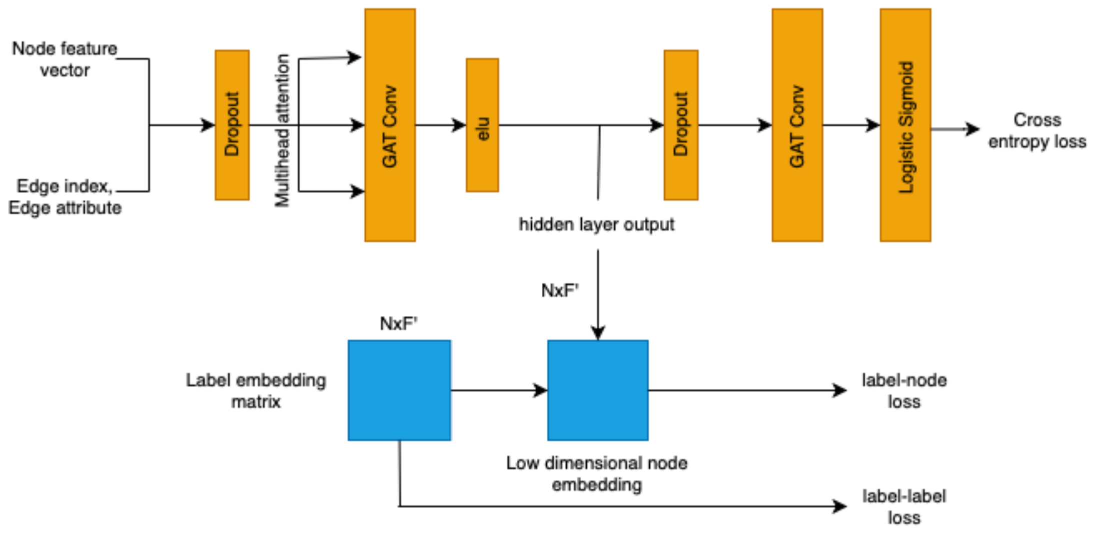

Figure 3.3 Network architecture for Multi-Label Graph Attention Network (ML-GAT)

The ML-GAT framework is presented in Figure 3.3. In the first layer of ML-GAT model, we use 8 attentional heads and concatenate the output from the heads (as used for Cora dataset in [3]) to get an output node feature embedding in the form of a $N\times F'$ tensor. In the second and final layer, we employ a single attention head followed by a logistic sigmoid activation. This output feature representation is used to compute the classification loss for training.

For node-label loss, all possible pairs of a node and its labels are constructed just like word pairs in the skip-gram model and a co-occurrence probability is calculated. Similarly, intra-label pairs are made for computing label-label loss co-occurrence probability. To decrease the summation requirement for calculating $P\left(z_{y_j}|\vec{h}_i\right)$ and $P\left(z_{y_j}|z_{y_i}\right)$ over all the labels, negative sampling technique with a sample size of 5 has been used. The node-label and label-label losses hence attained are given in 3.10 and 3.11 . Label-label, label-node and total loss are represented by L_{l-l}, L_{n-l} and L_{sum} respectively.

$$L_{n-l} += \left(-\log\sigma\left(\overrightarrow{Z_{y_i}}\overrightarrow{h_i}\right) - \sum_{k=1}^{K}\log\sigma\left(-\overrightarrow{Z_{y_k}}\overrightarrow{h_i}\right)\right) \quad (3.10)$$

$$L_{l-l} += \left(-\log\sigma\left(\overrightarrow{Z_{y_i}}\overrightarrow{Z_y}\right) - \sum_{k=1}^{K}\log\sigma\left(-\overrightarrow{Z_{y_k}}\overrightarrow{Z_{y_i}}\right)\right) \quad (3.11)$$

$$L_{sum} = \lambda_1 L_{n-l} + \lambda_2 L_{l-l} + l_{entropy} \quad (3.12)$$

The final objective to be optimized is given in 3.12 . $\lambda_1, \lambda_2 \in \mathbb{R}$, are the regularization hyperparameters to penalize the loss terms differently as per the optimization strategy used.

It is to be noted that as our models need to learn MLC and each training instance can belong to one or more labels i.e. belongs to {0,1}, we use a logistic sigmoid function instead of a softmax function on the output tensor. This is in accordance with prior methods used for MLC as is the case with GAT for inductive learning on PPI dataset [2]. For fair comparison with ML-GCN and ML-GAT and alignment with the datasets, the baseline GAT and GCN models we experiment with and report in this paper also use the same classification loss.

Making a separate label embedding matrix helps in getting label and node feature information together in a uniform space and makes computation of the loss terms easy. A label embedding matrix is generated to represent label embeddings in the same vector space as the node feature embedding. The dimensionality of this space should be large enough to capture all the information and prevent any loss of feature information due to a sudden shift from high dimensional input feature matrix to a very low dimensional network output feature matrix in models with few layers. At the same time, it shouldn't make the process computationally expensive. As input node vectors are of high dimensionality and the output embedding of the GAT has a low dimensionality, it was advised in [2] that the label embedding matrix have dimensions same as the node features before the last graph attentional layer (output of the 2nd last GAT layer).

3.2.3 Loss computation and back propagation

Cross entropy loss is computed between the output of the GAT network and train set labels. Node-label loss and label-label loss are calculated using a randomly generated label embedding matrix and output of the second last GAT layer. λ_1 and $\lambda_2$are used as regularisation parameters for node-label loss and label-label loss respectively. Summation of cross entropy loss and regularized node-label and label-label loss gives loss sum, which is used for backpropagation. L2 regularization and dropout to both layers' inputs during training.

3.2.4 Testing phase

Our project uses micro F1 score for test accuracy measurement. Micro F1 score is the standard accuracy measurement used in most of the multi-label classification models. Micro F1 score is based on performing aggregation of all the classes to compute average metrics hence micro F1 score is more suitable for multi label classification. Standard MLC models like ML-GCN and MAGNET use micro F1 score as an accuracy metric.

CHAPTER 4

EXPERIMENTAL RESULTS

In this section, we report an empirical evaluation of our model ML-GAT against three strong previous approaches on multi-label graph-structured datasets. The results attained in our experiments surpass existing benchmarks across all three state-of-the-art models in consideration. In the following section, we summarize our experimental setup, results and present a brief quantitative analysis of the relative performance of ML-GCN and ML-GAT models with variation of three GNN hyperparameters.

4.1 Dataset

For the purpose of this research, we use multi-label graph-structured datasets Yeast and Facebook. These two datasets have been selected because of the considerable difference in their application domain, topological structure as well as data sizes. Facebook is a social network dataset consisting of nodes representing the Facebook user and the edges between the nodes represents friendship relations between the nodes. The goal is to identify the 'social circles' each user belongs to. In the Yeast dataset nodes represent genes in the form of micro-array expression data and edges indicate gene interactions, the graph represents protein interaction in the yeast organism. The task is to predict the cellular localization sites of proteins based on genes and their interactions. Summarization of the characteristics of the datasets used in the experiments is presented in Table 4.1.

Datasets	Nodes	Edges	Features	Classes
Facebook	710	56824	480	46
Yeast	1240	4466	831	13

Table 4.1 Summary of the characteristics of the datasets

4.2 Baseline

We apply a two-layer GAT model like in the original paper ("Graph attention networks", Veličković, Petar, et al.). Further, we also reproduce results on facebook and yeast using GCN and ML-GCN. Dropout with p = 0.5 is applied to both layers' inputs as well as to the normalized attention coefficients for all the models. We train the models for 200 epochs and training size of 0.14 percent of the original dataset size. Reproduced baseline results are shown in table 4.2 .

Models	**Facebook**	**Yeast**
GCN	60	64.94
GAT	68.65	53.179
ML-GCN	71.64	64.78

Table 4.2 F1-score (test accuracy) of GCN, GAT and ML-GCN

4.3 Methods in Comparison

We perform experiments on the proposed ML-GAT model. Graph Convolutional Networks (GCN) and Graph Attention Networks(GAT) are well established graph neural network frameworks. The model ML-GCN was introduced in the paper [2]. During the course of experimentation over this model, we discovered hyperparameter combinations which could further optimize the original ML-GCN model. Hence, besides first reproducing the ML-GCN experiments as presented by the authors, we performed hyperparameter optimization and found directions that led to significant performance enhancement. The four graph neural networks in comparison are:

1. Graph Convolutional Network[1], that directly uses both node and graph topological information to perform semi-supervised learning on graph-structured data.
2. Graph Attention Network[3], a neural network architecture that employs self-attentional layers to operate on graph-structured data. It improves on the GCN models.
3. ML-GCN[2], a GCN based semi-supervised learning model tackling multi-label classification adapted for our experiments.
4. ML-GAT, an approach to augment GAT further for multi-label classification.

4.4 Experimental Settings

For fair comparison with previous literature, we use a two-layer network for all four models. We set the random seed to 42 and randomly shuffle the dataset before making the train, test and validation data splits. We use a fixed validation set for hyperparameter optimization. We chose the same test and validation dataset splits as in [2] with a train split of 20% and 30% for Facebook and yeast datasets respectively. Unless otherwise specified, we train the ML-GCN and ML-GAT models on Facebook using a middle layer dimensionality of 64 with 150 test, 100 validation and 142 train nodes. For yeast, we use a 128 unit middle layer and 500 test, 300 validation and 372 training nodes.

We optimize hyperparameters on both datasets individually and train all models for a maximum of 350 epochs using Adam SGD optimizer [5]. Glorot weight initialization as per [4] has been used with a weight decay of 5e-4. For our experiments, we use a learning rate of 0.01 and 0.005 throughout for Facebook and yeast respectively. Other than the additional experiments over dropout, we use a dropout of 0.2 for both datasets. Number of negative samples are set to 5 and to 0.25 as in paper [2]. During the course of experimentation over this model, we discovered hyperparameter combinations which could further optimize the original ML-GCN model. Hence, we performed hyperparameter optimization and found directions that led to significant performance enhancement. To make a fair comparison of ML-GCN and ML-GAT with their baselines GCN and GAT, unless otherwise stated, all reported results are from experiments performed using our optimized hyperparameters. Micro-F1 score (in percentage) which is a standard for MLC has been used as the evaluation metric for all experiments.

4.5 Overall Results

We perform experiments on the proposed ML-GAT model. Graph Convolutional Networks(GCN) and Graph Attention Networks(GAT) are well established graph neural network frameworks. We use these as baselines for comparison with the other two models. Results for the baselines GCN and GAT are produced by our learning strategy on these datasets

Our overall performance results are summarized in Table 4.3. In the table, the metrics for GCN*[2] and MLGCN*[2] have been reused from the ML-GCN paper Table 2 [2] to draw comparison. It is to be noted that we observe an increase of 12.63% and 10.09 % from the GCN* F1 values on Facebook and yeast respectively. In our experiments, MLGCN achieves an improvement of 1.53% over GCN for Facebook and 0.75 % for yeast. MLGAT outperforms the GAT by 4.68% and 0.5% on Facebook and yeast respectively. We also point out that MLGAT outperforms MLGCN for Facebook by a margin of 2.57%. The opposite is true for Yeast where MLGCN exceeds MLGAT performance by 1.94%.

Models	Facebook	Yeast
GCN*[2]	58.13	63.16
GCN	70.76	73.25
GAT	70.18	71.56
ML-GCN*[2]	59.85	66.06
MLGCN	72.29	74.0
MLGAT	74.86	72.06

Table 4.3 Summary of results in terms of micro-F1 score (in percentage) for Facebook and Yeast datasets

4.6 Comparative Study

4.6.1 Study based on dropout

Dropout	Facebook		Yeast	
	ML-GAT	ML-GCN	ML-GAT	ML-GCN
0.1	74.93	74.85	71.63	74.18
0.2	76.35	74.27	71.23	73.98
0.3	75.08	73.04	69.24	74.1
0.4	74.16	73.9	68.54	73
0.5	74.1	74.49	61.54	72.5
0.6	73.88	74.64	54.72	71.61

Table 4.4 F1 score obtained for various dropouts on Facebook and Yeast using ML-GAT and ML-GCN models

The effect of variation of layer dropout is presented in figure 4.1, and table 4.4 contains the F1 score obtained for various dropouts on Facebook and Yeast using ML-GAT and ML-GCN models. The same dropout is applied on both the layers of the 2 models. We observe sharp changes in performance as the value of dropout is increased from 0.1 to 0.6 for both models and datasets. Paper [2] used a dropout of 0.6 for these datasets. During the course of our experiments, we realized that using a lower dropout value such as 0.2 plays a key role in the performance increase and we use this value for all our experiments.

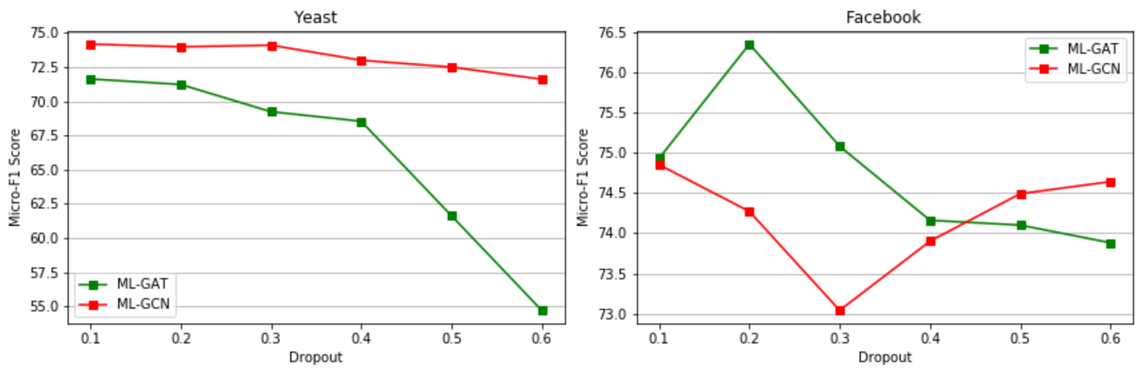

Figure 4.1 Influence of layer dropout variation on test set classification performance. We show the results of our experiments on both the standard ML-GCN model and proposed ML-GAT model.

4.6.2 Study based on λ_1 and λ_2

Lambda (λ_1,λ_2)	Facebook		Yeast	
	ML-GAT	ML-GCN	ML-GAT	ML-GCN
0.01,10	70.69	69.62	70	72.22
0.1,1.0	73.14	71.23	70.72	74.15
0.25,0.5	76.62	74.12	71.29	74.17
0.5,0.25	77.31	73.37	72.93	74.51
1.0,0.1	77.53	75.22	72.15	74.42
10,0.01	78.09	76.4	73.46	74.62

Table 4.5 F1 score obtained for various lambda1 and lambda2 pairs on Facebook and Yeast using ML-GAT and ML-GCN models

Table 4.5 contains the F1 score obtained for various lambda1 and lambda2 pairs on Facebook and Yeast using ML-GAT and ML-GCN models, and in figure 4.2, we investigate the combined influence of varying the regularisation hyperparameters and (penalties associated with label-node and label-label losses) on overall model performance. and were varied together in such a manner that increased as decreased to observe the relative significance of one with respect to the other. It is to be noted that the maximum accuracy attained for both datasets and models in all our experiments is for the combination of λ_1=10.0 and λ_2=0.01. These values are 73.46%

for ML-GAT and 74.64% for ML-GCN on yeast and 78.09% for ML-GAT and 76.4% for ML-GCN on Facebook dataset.

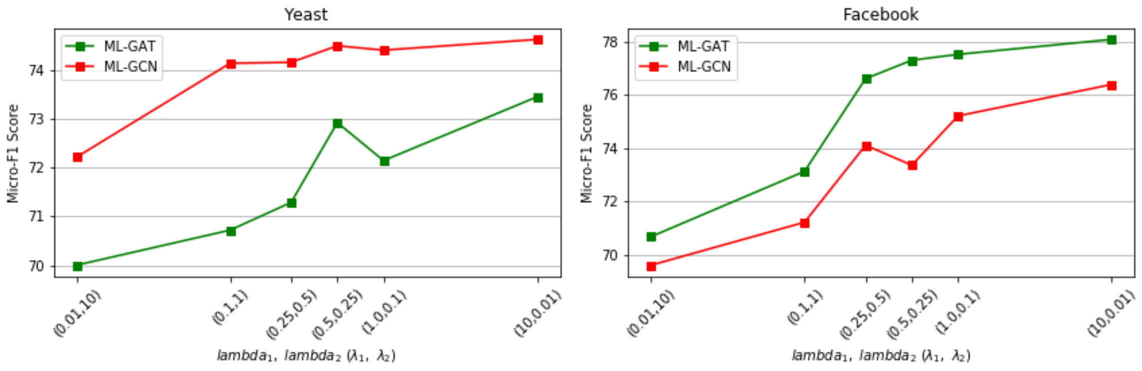

Figure 4.2 These graphs show classification performance over variation of regularization hyperparameters: and for both MLGCN and MLGAT. is the parameter associated with label-node loss and is associated with the label-label lose

In (3.12), and are used to weight the regularization loss terms   and   respectively. They control the amount of attention that the learning process should pay to the penalty or how much to penalize the model based on the values of the loss terms. If the penalty is too high, the model will underestimate the respective loss term while, if the penalty is too low, the model will lay higher emphasis and overestimate the concern addressed by the respective loss term. The values of 1and 2 corresponding to the best performing model show that >> and the model performs best when trained with higher penalty on label-node loss and very low penalty on label-label loss. This in effect proves the hypothesis that the GCN and GAT models inherently fail to capture label-label correlations because they treat each label individually and such inter-label correlations when added and emphasized, as in ML-GCN and ML-GAT for MLC result in significant increase in the Micro F1 measure. This hypothesis has previously been implied in the paper [2] (MLGCN) through the comparison over experiments on models Partly ML-GCN and ML-GCN which shows that the calculation of label-label loss in training improves the performance of the model as compared to the exclusion of the same from the objective function.

4.6.3 Study based on training size

Training size	Facebook		Yeast	
	ML-GAT	**ML-GCN**	**ML-GAT**	**ML-GCN**
10%	73.2	67.49	69.46	71.16
20%	76.35	74.27	70.4	73.33
30%	74.78	72.16	71.23	73.43
40%	75.64	71.19	71.97	74.95
50%	75.43	72.41	72.23	73.35
60% for facebook 58% for yeast	75.78	72.11	72.66	73.37

Table 4.6 F1 score obtained using different training dataset size on Facebook and Yeast using ML-GAT and ML-GCN models

An analysis of the influence of training size on test accuracy can be seen in Figure 4.3 and table 4.6 contains the F1 score obtained for various training dataset size on Facebook and Yeast using ML-GAT and ML-GCN models. In all these experiments, the test set is fixed at 150 nodes for Facebook and 500 nodes for yeast. Hence, the reported test set F1-scores shown on the graphs are comparable. For training sizes greater than 0.4, we include the validation set in the train set and train without validation. All other hyperparameters remain fixed . For Facebook, we observe the best performance with a training set size of 20% consistently for both ML-GCN and ML-GAT and hence use it for all experiments reported in the paper. In yeast, the F1-score more or less increases as we increase the number of training instances. It can be observed that for Facebook, ML-GAT consistently outperforms ML-GCN while the opposite is true for Yeast.

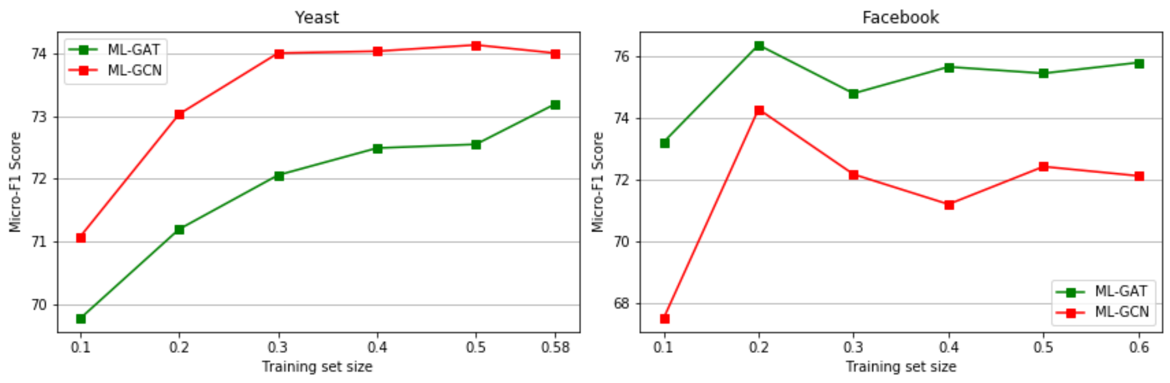

Figure 4.3 Influence of training set sizes on test set performance for both ML-GCN and ML-GAT in our experiments.

4.6.4 Study based on hidden layer

The number of layers as well as the number of hidden units play an important role in the performance of the model. With fewer number of layers the difference between the second last layer dimensions and the last layer dimension could be vast, resulting in possible hidden feature loss. Hence the number of layer and hidden units should be carefully selected in accordance with the dataset size. Large numbers of layers also have a negative impact because they can result in excessive mixing of features leading to loss in difference between different label features. Because the size of datasets facebook and yeast are small (710 and 1240 nodes respectively) two layered ML-GAT architecture is considered suitable for them. Further number of hidden layers should also be in accordance of the dataset size, because of the smaller size of facebook dataset hidden layer size of 64 is suitable of facebook and since yeast has comparatively greater size than facebook, model performs better with hidden layer size of 128 for yeast. This could be seen in table 4.7 as well.

Hidden Layer size	**Facebook**	**Yeast**
64	74.86	71
128	73.5	71.23
256	72.31	66.37

Table 4.7 F1 score obtained using different hidden layer size on Facebook and Yeast using ML-GAT

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

This paper examines the relative performances of the 4 models in consideration and gives key insights into their behaviour with respect to multi-label node classification. By setting new benchmarks and providing a detailed experimental analysis over the factors that affect these models for a multi-label semi-supervised learning scenario, this paper will serve as a benchmark and starting point for future research in this area on these datasets as MLC for graph-structured data is till date relatively unexplored. We emphasise on the relevance and importance of the label-label correlation which seems to capture information pivotal in affecting performance for the case of multiple labels. This turns out to be true for both GAT and GCN baselines which show an increase of ~1% consistently over all experiments.

5.2 Future Work

A possible direction for future work could be performing these experiments over large-scale inductive learning tasks like MLC on the protein-protein interaction dataset (PPI), which contains multiple graphs. While we cannot make inferences about the scalability of this approach as of now, it is worth pointing out that GAT has the following inherent features which should make ML-GAT better suited for large-scale datasets like PPI:

1. To reduce the storage complexity to linear in the number of nodes and edges, a sparse matrix version of the GAT layer could be used. This enables the execution of GAT models on larger graphs.
2. The attention based neural message passing mechanism doesn't depend on access to the entire graph structure upfront, making it apt for inductive learning datasets like PPI where nodes, edges, their features or labels may be missing at the time of model training.

CHAPTER 6

BIBLIOGRAPHY

6.1 Research literature

[1]. Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 **(2016).**

[2]. Gao, Kaisheng, Jing Zhang, and Cangqi Zhou. "Semi-Supervised Graph Embedding for Multi-Label Graph Node Classification." International Conference on Web Information Systems Engineering. Springer, Cham, **2019**.

[3]. Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 **(2017).**

[4]. Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the thirteenth international conference on artificial intelligence and statistics. **2010**.

[5]. Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 **(2014)**.

[6]. Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. **2014.**

[7]. Yang, Zhilin, William W. Cohen, and Ruslan Salakhutdinov. "Revisiting semi-supervised learning with graph embeddings." arXiv preprint arXiv:1603.08861 **(2016).**

[8]. Gori, Marco, Gabriele Monfardini, and Franco Scarselli. "A new model for learning in graph domains." Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.. Vol. 2. IEEE, **2005**.

[9]. Bruna, Joan, et al. "Spectral networks and locally connected networks on graphs." arXiv preprint arXiv:1312.6203 **(2013)**.

[10]. Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." Advances in neural information processing systems. **2016**.

[11]. Duvenaud, David K., et al. "Convolutional networks on graphs for learning molecular fingerprints." Advances in neural information processing systems. **2015**.

[12]. Atwood, James, and Don Towsley. "Diffusion-convolutional neural networks." Advances in neural information processing systems. **2016**.

[13]. Monti, Federico, et al. "Geometric deep learning on graphs and manifolds using mixture model cnns." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. **2017**.

[14]. Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." Advances in neural information processing systems. **2017**.

[15]. Akujuobi, Uchenna, et al. "Collaborative graph walk for semi-supervised multi-label node classification." arXiv preprint arXiv:1910.09706 **(2019)**.

[16]. Kong, Xiangnan, Xiaoxiao Shi, and Philip S. Yu. "Multi-label collective classification." Proceedings of the 2011 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, **2011**.

[17]. Zha, Zheng-Jun, et al. "Graph-based semi-supervised learning with multiple labels." Journal of Visual Communication and Image Representation 20.2 **(2009)**: 97-103.

[18]. Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. **2013**.

[19]. Pal, Ankit, Muru Selvakumar, and Malaikannan Sankarasubbu. "Multi-Label Text Classification using Attention-based Graph Neural Network." arXiv preprint arXiv:2003.11644 **(2020)**.

[20]. Shi, Min, et al. "Multi-Label Graph Convolutional Network Representation Learning." arXiv preprint arXiv:1912.11757 **(2019)**.

[21]. Yang, Zhilin, William W. Cohen, and Ruslan Salakhutdinov. "Revisiting semi-supervised learning with graph embeddings." *arXiv preprint arXiv:1603.08861* (2016).

[22]. Gong, Shunwang. *Geometric Deep Learning*. Diss. Imperial College London, 2018.

[23]. Zheng-Jun Zha, Tao Mei, Jingdong Wang, Zengfu Wang and Xian-Sheng Hua, "Graph-based semi-supervised learning with multi-label," *2008 IEEE International Conference on Multimedia and Expo*, Hannover, 2008, pp. 1321-1324, doi: 10.1109/ICME.2008.4607686.

[24]. Fout, Alex, et al. "Protein interface prediction using graph convolutional networks." *Advances in neural information processing systems*. 2017

[25]. Zitnik, Marinka, Monica Agrawal, and Jure Leskovec. "Modeling polypharmacy side effects with graph convolutional networks." *Bioinformatics* 34.13 (2018): i457-i466.

[26]. Yu, Bing, Haoteng Yin, and Zhanxing Zhu. "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting." *arXiv preprint arXiv:1709.04875* (2017).

[27]. Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016.

[28]. Narayanan, Annamalai, et al. "graph2vec: Learning distributed representations of graphs." *arXiv preprint arXiv:1707.05005* (2017).

[29]. Wang, Daixin, Peng Cui, and Wenwu Zhu. "Structural deep network embedding." *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016

[30]. Tang, Jian, et al. "Line: Large-scale information network embedding." *Proceedings of the 24th international conference on world wide web*. 2015.

6.2 Tutorials and articles

- http://petar-v.com/GAT/
- https://towardsdatascience.com/https-medium-com-aishwaryajadhav-applications-of-graph-neural-networks-1420576be574
- https://blog.fastforwardlabs.com/2019/10/30/exciting-applications-of-graph-neural-networks.html
- https://medium.com/@ODSC/a-brief-survey-of-node-classification-with-graph-neural-networks-fa02aff024e4
- https://pytorch-geometric.readthedocs.io/en/latest/notes/create_gnn.html
- https://www.youtube.com/watch?v=iDulhoQ2pro
- https://towardsdatascience.com/a-gentle-introduction-to-graph-neural-network-basics-deepwalk-and-graphsage-db5d540d50b3
- https://medium.com/stellargraph/knowing-your-neighbours-machine-learning-on-graphs-9b7c3d0d5896
- https://tkipf.github.io/graph-convolutional-networks/
- https://towardsdatascience.com/hands-on-graph-neural-networks-with-pytorch-pytorch-geometric-359487e221a8

- https://medium.com/syncedreview/pytorch-geometric-a-fast-pytorch-library-for-dl-a833dff466e5

- https://codesachin.wordpress.com/2016/07/03/a-small-and-easy-introduction-to-transductive-learning/

Btech Thesis

ORIGINALITY REPORT

8%	7%	6%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.imperial.ac.uk Internet Source	2%
2	"Web Information Systems Engineering – WISE 2019", Springer Science and Business Media LLC, 2019 Publication	2%
3	towardsdatascience.com Internet Source	1%
4	openreview.net Internet Source	1%
5	www.repository.cam.ac.uk Internet Source	1%
6	Submitted to Delhi Technological University Student Paper	1%
7	cs.unm.edu Internet Source	1%
8	academic.oup.com Internet Source	1%