1-16-2020

# Convolutional Neural Network for Link Prediction Based on Subgraphs in Social Networks

Kumaran Ragunathan
*University of Windsor*

# Convolutional Neural Network for Link Prediction Based on Subgraphs in Social Networks

by

**Kumaran Ragunathan**

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
**University of Windsor**

Windsor, Ontario, Canada
2020

**Convolutional Neural Network for Link Prediction Based on Subgraphs in Social Networks**

by

**Kumaran Ragunathan**

APPROVED BY:

---

A.Hussein

Department of Mathematics and Statistics

---

J.Lu

School of Computer Science

---

Z.Kobti, Advisor

School of Computer Science

January 16, 2020

# Declaration of Co-Authorship/Previous Publication

1. Co-Authorship

   I hereby declare that this thesis incorporates material that is the result of research conducted under the supervision of Dr. Ziad Kobti. In all cases, the key ideas, primary contributions, experimental designs, data analysis and interpretation, were performed by the author, and the contribution of co-authors was primarily through the proofreading of the published manuscripts. Kalyani Selvarajah contributed in discussion of concept and publication.

   I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

   I certify that, with the above qualification, this thesis, and the research to which it refers, is product of my own work.

2. Previous Publication

   This thesis includes the paper that has been accepted in a peer reviewed conference, as follows:

| Thesis Chapter | Publication title/Full citation | Publication status |
|---|---|---|
| Chapter [3] | K. Ragunathan, K. Selvarajah, and Z. Kobti. Link Prediction by Analyzing Common Neighbors Based Subgraphs using Convolutional Neural Network. 24th European Conference on Artificial Intelligence, 2020. | accepted for publication |

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

3. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis. I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

# Abstract

Link Prediction (LP) in social networks (SN) is referred to as predicting the likelihood of a link formation in SNs in the near future. There are several types of SNs that are available such as human interaction network, biological network, protein-to-protein interaction network, and so on. Earlier LP researches used heuristics methods, including Common Neighbors, Resource Allocation, and many other similarity score methods. Even though heuristics methods perform better in some types of SNs, their performance is limited in other types of SNs. Finding the best heuristics for a given type of SN is a trial and error process. Recent state-of-the-art research, WLNM and SEAL showed that with deep learning techniques and subgraphing, the heuristics selection could be automated and increase the accuracy of LP. However, WLNM and SEAL have some limitations and still having performance lack in some types of SNs. The objective of this paper is to introduce a novel framework that overcomes the limitations of state-of-the-art methods and improves the accuracy of LP over various types of social networks. We propose a Link Prediction framework called PLACN that analyzes common neighbors based subgraphs using deep learning technique to predict links. PLACN is equipped with two new algorithms that are a subgraph extraction algorithm that efficiently extracts common neighbors of targeted nodes and a proposed new node labeling algorithm based on hop number and average path weight that creates consistent node orders over subgraphs. In addition to the algorithms, we derived a formula based on network properties to find an optimal number node for a given SN. PLACN converts the LP problem into an Image Classification problem and utilizes a Convolutional Neural Network to classify the links. We tested the proposed PLACN on seven different types of real-work networks and compared the performance against heuristics, latent methods, and state-of-the-art methods. Our results show that PLACN outperformed the compared Link Prediction methods while reaching above 96% AUC in tested benchmark social networks.

# Dedication

I would like to dedicate this thesis to my family.

In memory of my father Ragunathan Subramaniyam

To my mother Ranjiny Ragunathan With love and eternal appreciation

# Acknowledgements

There are many people to whom I would like to acknowledge for their help and support for my journey of the master thesis.

First and foremost I would pay my gratitude to my supervisor Dr. Ziad Kobti. Under his guidance, I had enjoyed a lot working on my research work. It was a great pleasure to work and discuss with him. Without his support, this won't have been possible. I would also like to appreciate the amount of time he invested in me, the funding he provided and also the knowledge he shared with me.

In addition to this, many thanks to my committee members Dr. Jianguo Lu and Dr. Abdulkadir Hussein for their valuable time and their support. I would like to express my appreciation to Mrs. Gloria Mensah, Mrs. Karen Bourdeau, Mrs. Melissa Robinet and Mrs. Christine Weisener who always supported me when I needed assistance in various academic issues.

I would like to thank my mother for her counsel and the sympathetic ear. She is always there for me and support me in all possible ways. Next, I am also very thankful to my friends for their moral support and listening to my problems for long hours. Finally, I would like to thank God for unconditional love.

Kumaran Ragunathan

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Link Prediction (LP) is a problem in Social Network Analysis that focuses on predicting links that are going to appear in the near future [25]. Many heuristics methods have been proposed in early research that finds similarity between targeted nodes and predicts link existence based on the score[28, 25, 2]. However, heuristics methods performed well in some specific social networks, not others. In later researches, latent methods are proposed to improve the accuracy of link prediction [3, 13, 36, 19, 33]. However, those methods could be able to improve on specific types of social networks. The new state-of-the-art method [43] showed that subgraphing improves link prediction significantly.

## 1.1 Background

### 1.1.1 Social Network Analysis

Social Network Analysis is the study of social relations (e.g., friendship, co-workers) among a set of actors (e.g., students, patients, workers ) [34]. Therefore, the information on a social network is usually presented as a network graph. The social networks can be converted into graphs where actors such as a person, organization

| **Friend** | **Friends with** |
|---|---|
| John | Peter, Lucy |
| Peter | John, David, Lucy |
| Lucy | John, Osman, Peter |
| Osman | Lucy |
| David | Peter |

Table 1.1: Example friendship network.

are represented as nodes, and their relationship represented as edges. For example, consider a friendship network as given below in Example 1.1.The example shows how the network is converted into a graph.

**Example 1.1**: To illustrate the social network analysis, consider a friendship network with five people John, Peter, Lucy, Osman, and David, where people are connected through friendship. Table 1.1 shows people and the friendship between them. Here, we consider five people and show who are connected.Now the relationship shown in Table 1.1 is converted into a graph where people are nodes and their relationships are edges. The graph is shown in Figure 1.1 where nodes are people and edges are their friendships.

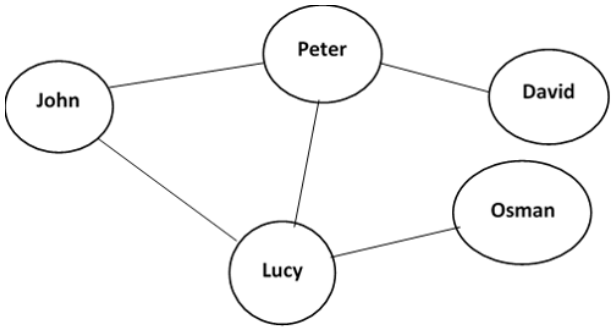


Figure 1.1: Friendship network graph representation

Once the social network converted into graph as shown in Figure 1.1, the graph can be further converted as adjacency matrix. In which, row and columns of the matrix

| | **John** | **Peter** | **Lucy** | **Osman** | **David** |
|---|---|---|---|---|---|
| **John** | 0 | 1 | 1 | 0 | 0 |
| **Peter** | 1 | 0 | 1 | 0 | 1 |
| **Lucy** | 1 | 1 | 0 | 1 | 0 |
| **Osman** | 0 | 0 | 1 | 0 | 0 |
| **David** | 0 | 1 | 0 | 0 | 0 |

Table 1.2: Adjacency matrix of the example friendship network.

indicates all nodes of the graph. If they have connected, then the corresponding cell value will be 1 and if the nodes don't have connection, it will be 0. The table 1.2 is an illustration for the adjacency matrix for the Example 1.1.

In the process of working in this field, network have categorized and investigated as a set of distinctive problems in social networks

- **Link Prediction** - Link prediction is to predict whether there will be links between two nodes based on the attribute information and the observed existing link information [25]. Link prediction not only can be used in the field of social network but can also be applied in other fields including bioinformatics, electronic commerce, recommendation system, the security field and hidden terrorist criminal gangs.

- **Community Detection** - The problem that community detection attempts to solve the identification of groups of nodes that are more densely connected to each other than to the rest of the network [31]. Detecting and analyzing the community structure of networks has led to important findings in a wide range of domains his problem refers finding similar group of nodes within the network graph.

- **Network Diffusion** - The study of network diffusion tries to capture the underlining mechanism of how events have propagated through complex networks

[20] whether the subject of interest is a virus spreading through some population or the spreading of some social movement some new fashion or innovation or it may be a marketing message it's spreading through an online social network.

- **Network Influence** - Network Influence is for analyzing the influencing manner among users and the spreading manner of influence based on social networking structure [11]. Network influence can be used in various disciplines including sociology to understand people's social behaviors, public services to provide a theoretical basis for public decision making and public opinion guidance and in terms of country, helpful to promote national security, economic stability, economic progress.

### 1.1.2 Link Prediction

Link Prediction (LP) is for a given snapshot of the network at a time, predicting the links that are most likely to form in the near future [25]. LP has many applications such as movie recommendation [12], friend recommendation [7, 41], metabolic network reconstruction [24], knowledge graph completion [17], and predicting protein-to-protein interactions [5]. The classic approach for link prediction is the heuristics method. Heuristic methods calculate similarity scores between nodes in a given network that yields the likelihood of link formation. The similarity scores are calculated based on attributes of the targeted nodes. These similarity scores can be categorized based on the degree of hop count needed for calculating the score. For example, Common Neighbors (CN)[28] and Preferential Attachment (PA)[8] are categorized as the first-order heuristic as they require first-hop information.

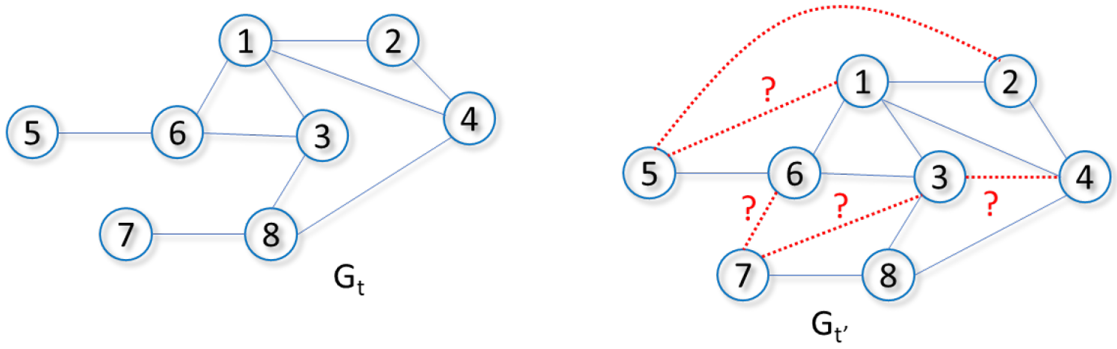


Figure 1.2: Network G for Link Prediction

Figure 1.2 illustrates a social network G with eight nodes and ten edges at time $t$. Link Prediction is aimed to predict potential links that are about to form at time $t'$. Possible links that can be formed in the future are denoted with a red dashed line in Figure 1.2.

### 1.1.3 Convolutional Neural Network

Convolutional Neural Network (CNN) is a type of Neural Network that is frequently used for Image Classification [18, 22, 30], Image Recognition [21, 14, 23], and Object Detection [32]. CNN consists of different layers to perform classification. The first layer is the input layer; after that, one or more convolutional layers and pooling layers can be present; finally, it consists of one or more dense layers and has an output layer. An example CNN is illustrated with its different layers in Figure 1.3.

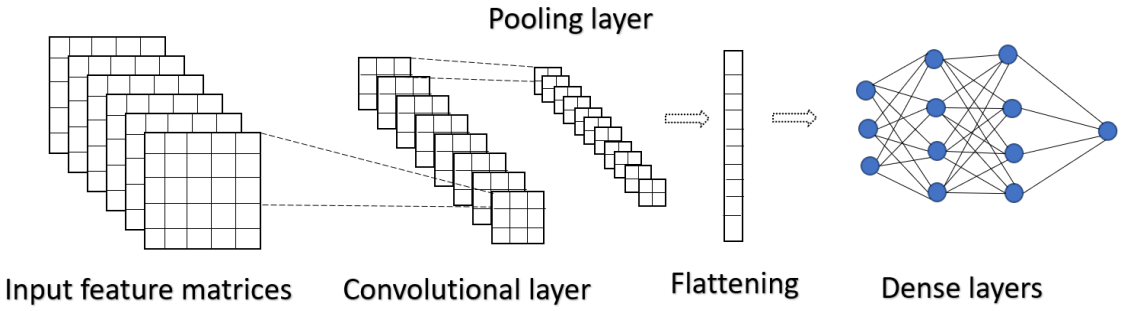


Figure 1.3: Convolutional Neural Network

The convolutional layer is special as it used to extract features from input matrices. Neurons in this layer are positioned as two-dimensional arrays and referred to as the activation map. Weights are arranged in a three-dimensional array known as the kernel. Height, width, and depth might be changed for different input sizes. There can be multiple kernels with different sizes, even for the same input matrix. Kernal slides through the input matrix and creates a set of activation maps. Figure 1.4 illustrates convolution operation on a input matrix.

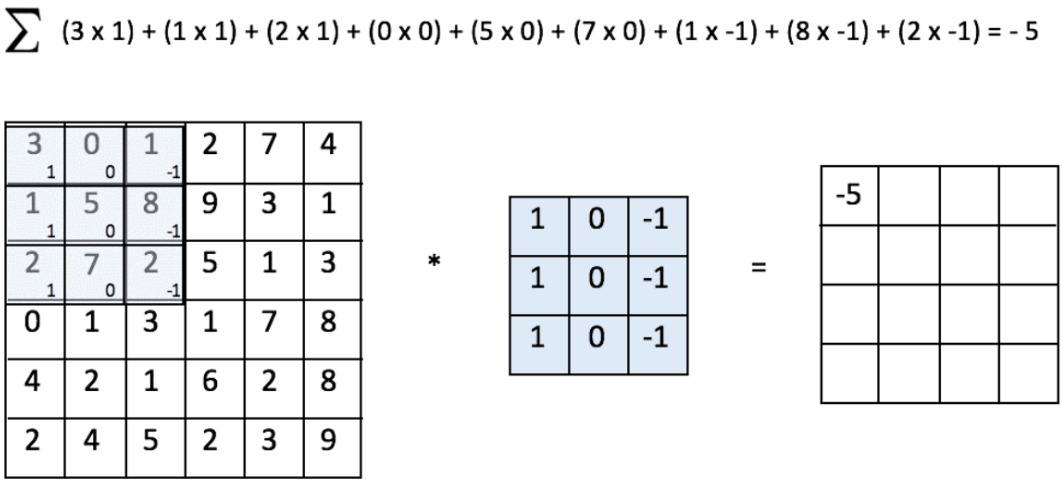


Figure 1.4: Convolutional layer

Convolutional layers reduce the weight optimization problem as they extract spatial features and feed them to neurons rather than connecting one neuron to each cell for the input matrix. In addition to the weights in kernels, a bias value is also added while creating an activation map. The activation value calculation by convolution layer can be expressed as follow in the equation 1.1

$$a^l_{j,k} = \sigma\left(b^{l-1} + \sum_{m=0}^{p}\sum_{n=0}^{q} w^{l-1}_{m,n} a^{l-1}_{j+m,k+n}\right) \tag{1.1}$$

where $a^l_{j,k}$ is the result activation value of the $k^{th}$ neuron in the $j^{th}$ row of the $l^{th}$ layer, $b$ is the bias shared among the layer, $w$ is the weight parameter of the $p \times q$

size kernel, and $\sigma(z)$ is the activation function.

The pooling layer is usually added just after a convolutional layer. Pooling later reduces dimensionality but not reduces depths or channels. Convolutional layer outputs feature maps as patches. Pooling layer kernels slides through these patches and creates reduced and summarized feature maps. There are three types of kernels used in CNN, which are Max Pooling, Min pooling, and Average pooling. Max pooling takes maximum values of the window while filter slides; Min pooling takes minimum value of the window while filter slides, and Average pooling takes an average of all value in the window while filter slides through. The most common type of pooling used in CNN is Max pooling. A simple example of Max pooling is illustrated in Figure 1.5.

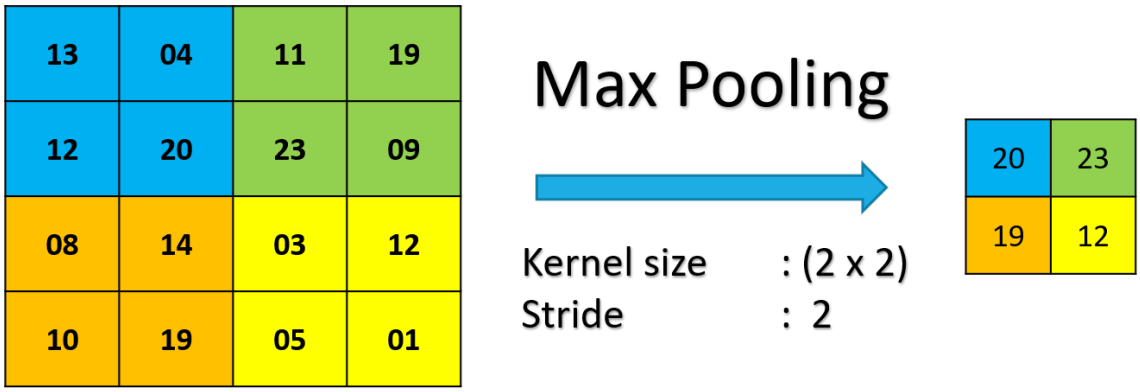


Figure 1.5: Max Pooling layer

Figure 1.5 represents how the dimensions are reduced, and the highlighted feature is preserved. There can be multiple sets of convolutional layers and pooling layers. After the convolutional layer and pooling layer, the result matrix is flattened and feed into a dense layer, which is usually a multi-layer perceptron network. Finally, the output layer will make the classification.

## 1.2 Problem Definition

We define a static Social Network as $G(V, E)$, a network snapshot at time $t$. We assume that the network snapshots in $G$ share the same vertex set.

Given graph $G = (V, E)$ at time $t$ represents a network snapshot, which contains a set of vertices $V$ indicating nodes, a set of edges $E$, indicating the relationships between two vertices or nodes. We consider an adjacency matrix $A_t$ with size of $(N \times N)$ to describe its static topological structure at time $t$, where $N = |V|$ is the number of unique vertices in the social networks. Since we consider weighted networks, the matrix $A$ has arbitrary edge weights that may change over time. Let assume an edge between nodes $i$ and $j$,that is, $(i, j) \in E_t$, with edge weight $(W_t)ij$, we have $(A_t)ij = (W_t)ij$ ; otherwise $(A_t)ij = 0$. Link Prediction focus at predicting the network snapshot $A_{t'}$ at timestamp $t'$ by observing the series of $A_t, A_{t'}$.

The problem is to predict the new edges $(E'_{t'} \cap E_t)$ which are likely to appear in time t'

## 1.3 Thesis Motivation

People are connected in real life through friendship or other relationships. People's network tends to grow as they introduce to each other. Social networks not only limited to people but also includes organizations, publications, biological network, traffic networks, and even protein-to-protein interaction network. Predicting the connections that are about to form in the future helps a lot to make life easier. Link Prediction in the biological network helps researchers to predict the behavior of species and reduces their research time.

Link Prediction is used in many applications such as Recommender Systems, Knowledge Graph Completion, Biological Web Completion, Terrorist Cell Identifi-

cation. Nowadays, the availability of data and increased computation power derived Artificial Intelligence and Deep Learning research to peek. Early researches in Link Prediction only utilizes heuristics methods, which used similarity scores to predict links. Even though heuristic methods show better results in particular types of social networks, they did not perform well in all types of social networks. This performance inconsistency leads to create new similarity scores and combine various similarity scores to create new link prediction methods to gain better results in Link Prediction.

Latent methods are created in order to improve accuracy in some types of social networks, but these methods showed lower accuracy in other types of social networks than basic heuristics methods. Finding the best Link Prediction method for a particular type of social network is often a trial and error process. State-of-the-art methods WLNM[42] and SEAL[43] proposed novel approached with a subgraphing method to solve performance inconsistency problems in various types of social networks. They improved the accuracy of Link Prediction in most types of social networks than heuristics and latent methods. However, they had some limitations in the algorithms such as inconsistency in node labeling and manual process to find the optimum size of the subgraph hop number.

The limitations of state-of-the-art methods and manual process motivated this research to solve and proposed a new methodology to overcome the limitations they had. This research aims to propose a novel framework that automates the selection of the best combination of heuristics for a given social network and improves the accuracy of Link Prediction by analyzing common neighbors. This framework solves the problem of manually finding the best method to get better accuracy of a given social network.

## 1.4 Thesis Statement

The objective of this research is to create a novel method of Link Prediction that can automatically learn the best combination of heuristics methods to improve the accuracy of social networks and not limited to certain types of social networks. Our aim solves the problem of the trial and error process of selecting the best method for a given type of social network.

Our proposed approach is to analyze not only targeted nodes of the link but also common neighbors of the targeted nodes. Common neighbors are highly influencing the formation of links. We design our framework to extract a subgraph that contains targeted nodes and common neighbors of links and then analyze the extracted subgraph to get information on link existence. We have proposed new algorithms to extract subgraphs and labeling the nodes. Our framework creates feature maps of extracted subgraphs to get information about the link presence. Convolutional Neural Network is known for image classification. We utilize this characteristic of CNN to solve the Link Prediction problem. Features map are the best candidates for CNN training. We convert the Link Prediction problem into the Image Classification problem and classify them with CNN.

We solve the problem of the trial and error process of finding the best method to solve link prediction on a given type of social network by proposing a novel framework that automatically adapts to a given social network and learns patterns to predict links.

## 1.5 Thesis Contribution

This thesis addresses the Link Prediction problem and proposes a novel framework named "Predicting Link by Analysing Common Neighbors"(PLACN) that improves the accuracy of Link prediction and adapts to various types of social networks. The proposed framework predicts the link by learning the enclosed subgraph of targeted nodes and their common neighbors.

PLACN framework consists of various steps to solve the Link Prediction problem. Social Network is given as input for the framework and framework processes the data and makes predictions. Each step is designed in a way that it solves limitations that state-of-the-art methods had and approached in a novel way. Five significant contributions made this proposed framework novel. The steps and contributions are listed below.

- Subgraph Extraction: New algorithm is proposed to extract subgraph that only focuses on common neighbors and targeted nodes of the link.
- Node labeling: New hop-number and weight hybrid based node labeling algorithm is proposed to label and order nodes in the subgraph to keep consistency among all subgraphs.
- Subgraphing factor $K$: New formula is derived and proposed to automatically calculate the optimum number of nodes in the subgraph for a given social network based on network properties.
- Feature matrices construction: Construct feature matrices as layers for a given subgraph to get information about targeted nodes and common neighbors.
- CNN: Constructs Convolutional Neural Network and train the CNN to classify positive links and negative links

## 1.6 Thesis Organization

The rest of the thesis/research work is organized in the following manner.

In chapter II, we discuss related work/literature review in the field of Link Prediction (LP) in a social network such as latent methods, state-of-the-art subgraph methods WLNM and SEAL.

In chapter III, We introduce our proposed Link Prediction framework PLACN which is a novel framework that adapts to various social network types. It not only analyses target nodes but also common neighbors of targeted nodes to select the best combination of heuristics for the given network. In the chapter, we give detailed description steps of PLACN framework and how it predicts the future links.

Chapter IV, We explain our experimental setup. This chapter presents the details of tested environmental setups, details of tested datasets, the configuration of the CNN used in PLACN, evaluation methods, and train test setups..

In Chapter V, In this chapter, we presented the test results of hueristics,latent, state-of-the-art methods, and compared with PLACN's performance on seven different types of social network datasets. It also presented the result and discussion on empirical verification of the proposed subgraphing factor $K$. This chapter consists of statistical significance test results to test PLACN's performance against state-of-the-art method SEAL's performance..

Chapter VI, this chapter concludes the research, explains insights received during the work and sets up the field of opportunities for the future work.

# Chapter 2

# Related Work and Literature Review

This chapter focuses on related works and researches used for background study, concept-building, and theoretical background of our thesis. We discuss and analyze works of literature that are relevant to Link Prediction(LP) problem, heuristics methods to solve LP, latent methods to solve LP, and state-of-the-art methods WLNM and SEAL.

Link Prediction is an active research field that contributes to many real-world applications such as Recommendation systems in social network applications [7, 41], website links[16, 26, 27], Co-author recommendations[6, 10, 15]. LibenNowell and Kleinberg (2007) drew a significant attention to Link Prediction in Social networks. Earlier researches focused on using topological features to calculate similarity scores and used them to predict links. The most commonly used similarity scores were Common Neighbors[28],Adamic Adar, Jaccard Coefficient [25], Katz index[2] , Resource Allocation and Pagerank[25]. These similarity scores were calculated for the targeted nodes, and a certain threshold is defined for a given social network. Then based on

the score and threshold, links were predicted. Later machine learning algorithms were used to predict links by classifying the links as links that will form in the near future and links that are not. The important researches in Link Prediction using machine learning methods are SVM [4], Decision Tree [38], and Random Walk [7]. The machine learning methods more focused on specific nodes than network topologies.

Latent methods are proposed to improve link prediction accuracy. The Stochastic Block Model (SBM)[3] creates blocks and assign nodes to the blocks. The link is predicted based on the relationship between blocks they assigned. However, SBM is computationally expensive and only performs well on certain types of social networks. Node-to-vector[13] is another approach in link prediction, which uses the word-to-vector technique and random walk to predict links. Large-scale Information Network Embedding (LINE)[36] is an embedding approach that combines first-order and second-order proximities to predict links. Variational graph auto-encoder (VGAE)[19] is a framework that uses unsupervised learning techniques to learn the pattern of graph structures data based on the variational auto-encoder.

Recently, a new subgraphing method WLNM [42], was proposed. The WLNM is a novel approach that extracts subgraph around both targeted nodes. The number of nodes for subgraph extraction is set manually, and similarity scores were calculated between targeted nodes. Nodes are labeled using the Palette-WL algorithm, which is a variant of WL to preserve the order. Adjacency matrix and similarity score vector are given to fully connected neural networks to predict links. WLNM had many drawbacks and limitations. SEAL [43] was proposed by Zhang et al. in 2018, which is also a subgraphing method. SEAL extracted subgraph based on the hop number given by the user. The subgraphs consist of all neighbors of targeted nodes at a given hop number and less. They calculated first-order, second-order, and high-order

heuristic scores to create a vector. The authors used their proposed Double Radius algorithm to order nodes. Finally, they feed the adjacency matrix and vector to graph neural networks to classify links. However, finding a suitable hop number for a given network is a trial and error process, and including all neighbor nodes n the subgraph creates a problem that hub nodes have a massive number of neighbor nodes even in low hop number. Another drawback is that their proposed node labeling algorithm gives the same labels to the common nodes which are in the same order. This effect creates an inconsistency in node order over subgraphs.

Based on the background study, related works, and limitations of the previous works, we proposed a novel framework to solve the link prediction problem. We described our proposed model in detail in Chapter 3.

# Chapter 3

# Proposed Approach

## 3.1 Introduction

Predicting Link by Analyzing Common Neighbors (PLACN) is the proposed novel framework that aims to increase accuracy of link prediction in social networks by analyzing common neighbors of targeted nodes. Another objective of this framework is to create a framework that can adapt to any type of social network and automate learning best combination of heuristics for given social network. We are introducing a new algorithm to extract subgraph and labeling the nodes in subgraph and using a CNN to classify positive and negative links. PLACN converts link prediction problem into an Image Classification problem with the new algorithms. PLACN utilizes the characteristic of CNN, which is best known for image classification. Another novelty is that past researches are focus on only the targeted nodes for link prediction, but PLACN is not only focusing on targeted nodes but also analyzing common neighbors between targeted nodes. As part of the framework we propose a new factor for subgraph size. We derived a formula to calculate subgraph size based on network properties. This process automates the optimization of subgraph calculation which reduces computational complexity. The architecture of PLACN is given in Figure:

3.1, which illustrates the process steps of link prediction.

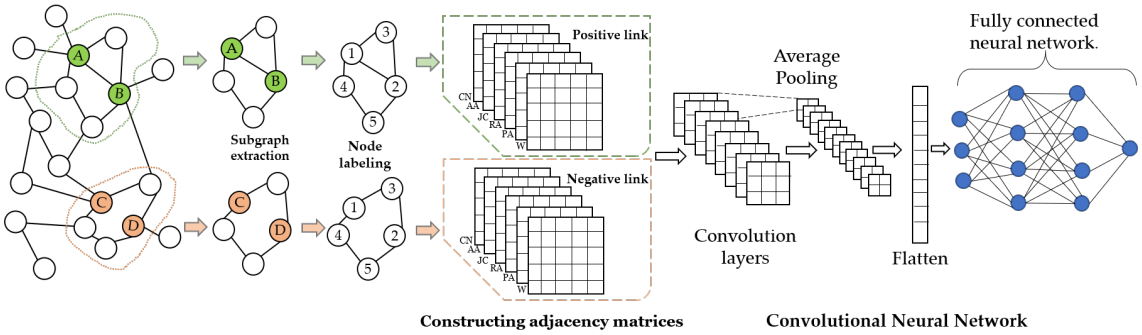


Figure 3.1: PLACN architecture

PLACN takes a social network as input and automatically creates a dataset for positive link class, positive link is a link that will present in the near future and negative link class, negative link is a link that will not present in the near future. Then we trains a CNN using data set to classify links. The PLACN frameworks consist of four main steps. The steps are listed below.

- Subgraph extraction
- Subgraph Node labeling
- Construction of feature matrices
- Convolutional Neural Network train and Classification

In addition to the steps, PLACN calculates subgraphing factor $K$ with the proposed formula. All four steps and sub graphing factor calculations are explained in detail in the following subsections.

## 3.2 Subgraph Extraction

PLACN aims to extract information not only from source and target nodes but also from neighbors of them. Usually, number of common neighbors is used as a metric

to predict the link. It illustrates the importance and influence of neighbors for an existence of a link. Studying the nodes around source and target will give more information of a link existence. WLNM first proposed the subgraph extraction method for link prediction[42]. WLNM extracts all neighbors at first hop of source and target nodes and selected first K nodes from them. Then to achieve more accuracy, they have increased hop numbers and increase the K. The process is a trial and error process for different social networks. Authors have tried different hop numbers and different $K$ to achieve an optimal number of hops and $K$s. In 2018, SEAL[43] was proposed, and authors extracted all neighbors' nodes for a given hop number and analyzed them to predict link.

Both WLNM and SEAL extracted all neighbor nodes of source and target nodes to create a subgraph. Hop number was a hyperparameter, and finding the optimal hop number is a manual process. Another limitation is that when the hop number is increased, the size of the subgraph grew exponentially and increases processing complexity. SEAL considers all neighbors and not limiting the number of nodes in the subgraph. This extraction method creates a drawback when subgraph extraction is applied to hub nodes or more influential nodes, hub nodes have a high number of neighbors, and even with hop count one, they tend to have a very high number of nodes in the subgraph. Authors of SEAL have stated this as a drawback in [43], and it limits the ability to test the SEAL framework with a high number of hops. SEAL tried only up to 3 hops to extract subgraphs.

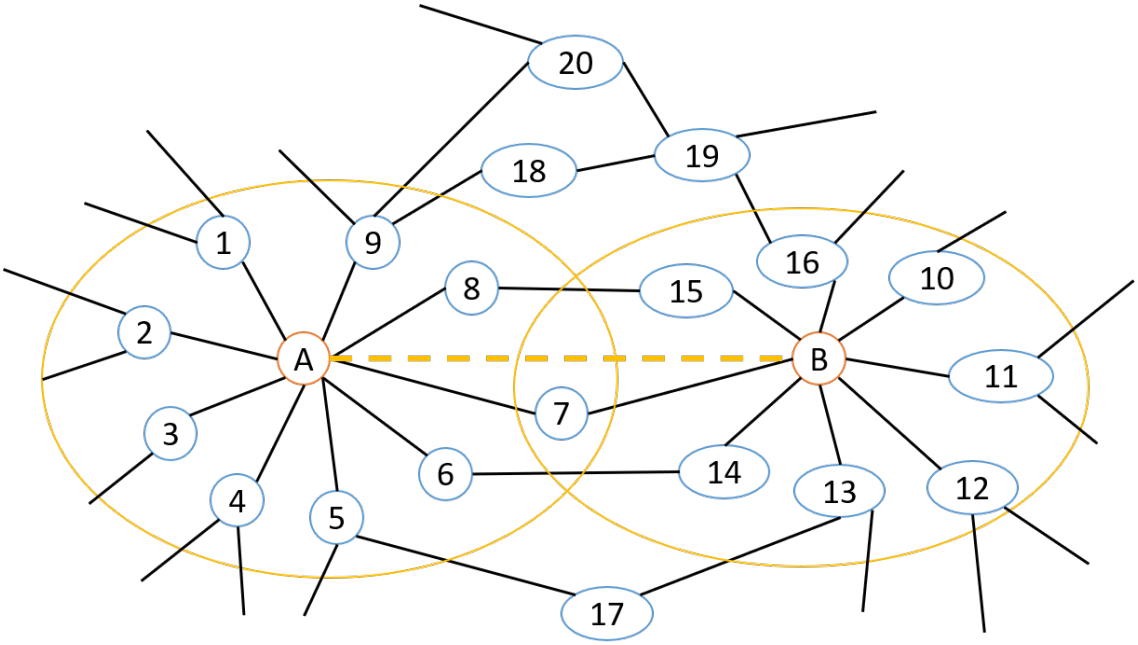


Figure 3.2: Subgraph extraction SEAL and WLNM method

Figure 3.2 illustrates the subgraph extraction of SEAL and WLNM frameworks. The nodes, which are extracted around the source and target node are indicated with a yellow circle. Out of 16 nodes included in the subgraph, only 9 of them are linked with both source and target nodes. Nodes 17, 18, 19, 20 are connected with target nodes but not included, so it leads to information loss.

PLACN proposes a new methodology to extract subgraph for a targeted link. Nodes that are common for both source and target node will impact both nodes for link existence. PLACN introduces a new number, sub graphing factor $K$. $K$ is the number of nodes in the subgraph. Subgraphing factor will vary for each social network, and it relies on the given graph properties. PLACN proposes a new algorithm for subgraph extraction based on common neighbors. The algorithm extracts common neighbors between source and target nodes in first-order neighbors and increases order until the number of nodes equal to or greater than the sub graphing factor.

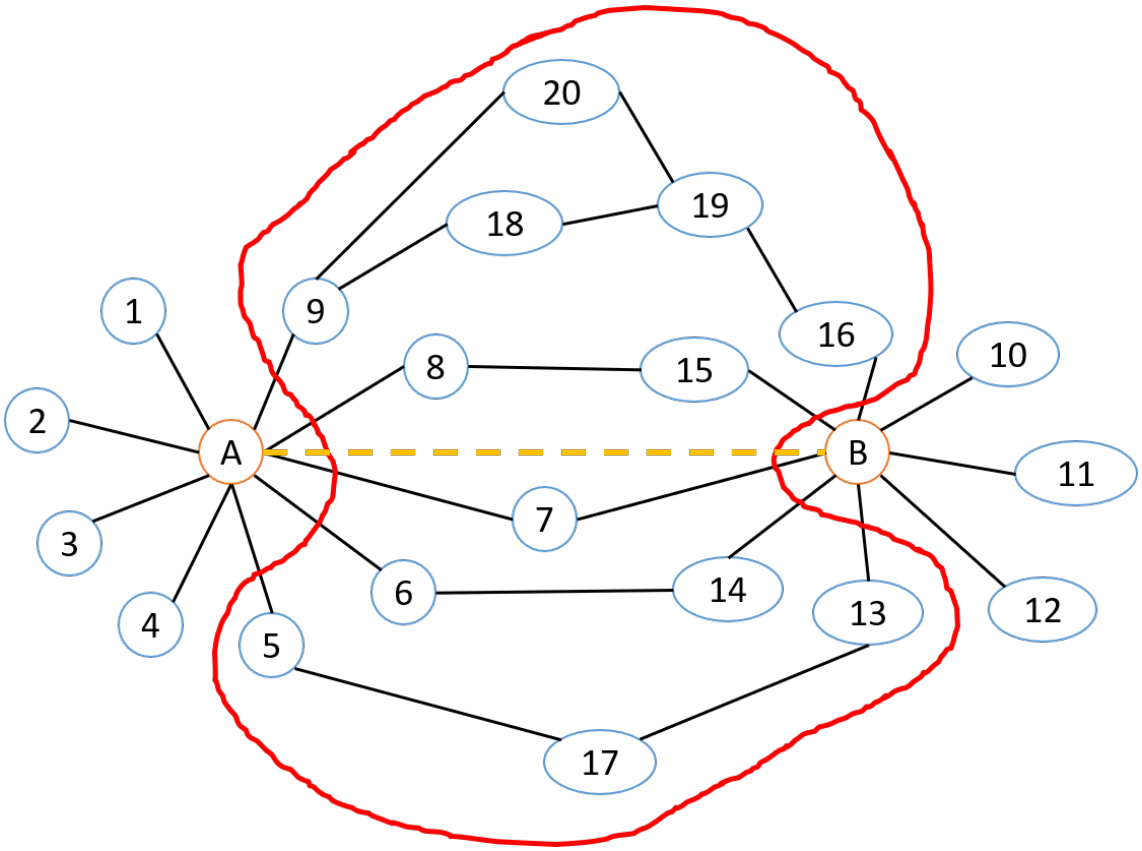


Figure 3.3: Subgraph extraction with proposed algorithm in PLACN

Figure 3.3 illustrates the extracted subgraph using the proposed algorithm 1 to the same graph as Figure 3.2. Sub graphing factor $K$ is considered as 13 in the given scenario. The resulting subgraph will have 13 nodes, and all of them are connected with both source node "A" and target node "B."

The proposed subgraph extraction algorithm only adds common nodes in each hop. The algorithm takes Target link $E_{ij}$, Graph $G(E, V)$, and Expected number of nodes in subgraph $K$ as inputs, and returns Subgraph $\langle S \rangle$ for the link $E_{ij}$. $\Gamma^h(i)$ represents set of neighbors from node $i$ at distance of $h$ hops. The subgraph extraction algorithm is given in Algorithm 1.

The resulting subgraph may contain more nodes than the given sub-graphing

**Algorithm 1** Subgraph Extraction

**Input**: Target link $E_{ij}$, Graph $G(E, V)$, and Expected number of nodes in subgraph $K$.
**Output**: Subgraph $\langle S \rangle$ for the link $E_{ij}$.

1: $N_K = \{i, j\}$
2: $N_{temp} = \{\}$
3: $h = 1 \leftarrow$ number of order
4: **while** $|N_K| < K$ **do**
5: $\quad N_{temp} = \Gamma^h(i) \cap \Gamma^h(j)$
6: $\quad N_K = N_K \cup N_{temp}$
7: $\quad h \leftarrow h + 1$
8: **end while**
9: $\langle S \rangle = subgraphG(N_K)$
10: **return** $\langle S \rangle$

factor $K$. The nodes will be ranked and labeled in the subgraph labeling algorithm. After that, if the number of nodes is greater than K, then excess nodes will be eliminated from the bottom rank. In case, a number of nodes are less than $K$, no new node will be added, and the resulting adjacency matrix will have 0 values for remaining nodes.

## 3.3 Subgraphing Factor $K$

Selecting an optimal number of nodes for a subgraph is a trivial task. A small number of nodes may not give enough information about link existence, and large subgraph size may lead to higher computational cost. The optimal number should able to provide enough information to predict links as well as reduce computation. In previous researches, this task is handled manually and was a trial and error process. PLACN framework is automating this process by calculating the sub graphing factor $K$ based on the given social network's properties.

For a given graph with $V$ vertices and $E$ number of edges, Average node degree

can be given by the following equation.

$$Avg.NodeDegree = \frac{2|E|}{|V|} \tag{3.1}$$

Network Density can be given as

$$NetworkDensity = \frac{2|E|}{|V|(|V|-1)} \tag{3.2}$$

The given properties are highly proportional to neighbors for a random node in a given social network. Since PLACN focuses on common neighbors, it takes the average node degree and considers high order neighbors. So the average node degree and its fraction of it will give an optimum number to get information to predict link. We formulate the equation based on the network properties and calculate the $K$ as follows.

$$\begin{aligned} K &\approx \text{AvgNodeDegree} + \text{ AvgNodeDegree} \times \text{NetworkDensity} \\ &\approx \text{AvgNodeDegree } (1 + \text{NetworkDensity}) \end{aligned}$$

$$K \approx \left\lceil \frac{2|E|}{|V|} \left(1 + \frac{2|E|}{|V|(|V|-1)}\right) \right\rceil \tag{3.3}$$

Since the resulting number might be a fraction, we take the ceiling of it. We have empirically verified that $K$ is an optimum number for subgraph with various social networks by changing $K$ and calculating the accuracy of it. The results and analysis of it discussed in the results and discussion section.

## 3.4 Subgraph Node Labeling

In this section, the importance of subgraph node labeling and the proposed node labeling algorithm will be analyzed. Extracted subgraphs will be considered as data to train the CNN in PLACN model. Machine learning models are always trained in

sequential order. Features of data should be in the same order for all data points. Neural networks are trained in order to optimize their weights. Inconsistent order of features will not help converge weights minimization of loss value during the training process. Subgraphs are converted into adjacency matrices and feed to CNN

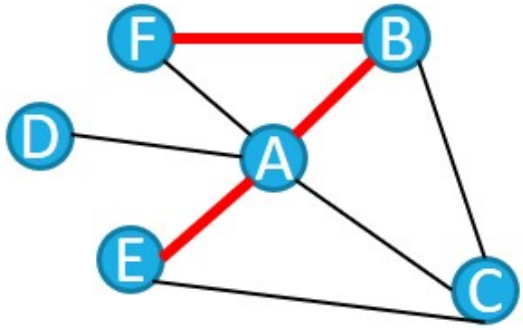


Figure 3.4: Simple Network with 6 nodes and positive links that are extracted are highlighted in red

An example can explain the importance of node labeling. Figure 3.4 shows a simple social network with six nodes A, B, C, D, E, F. Subgraph factor is taken as 3. Data for classes positive link and negative link will be generated by subgraph extraction using the proposed subgraph extraction algorithm 1. Positive links AB, BF, and AE are highlighted in red. Each positive link is a data record in a class. For links AB, BF, AE the extracted subgraphs will contain nodes {A, B, F}, {A, B, F}, {A, C, E} respectively.

| | A | B | F |
|---|---|---|---|
| A | | * | |
| B | * | | |
| F | | | |

A-B

| | A | B | F |
|---|---|---|---|
| A | | | |
| B | | | * |
| F | | * | |

B-F

| | A | C | E |
|---|---|---|---|
| A | | | * |
| C | | | |
| E | * | | |

A-E

Figure 3.5: Adjacency matrices of extracted subgraphs for links AB, BF and AE

Figure 3.5 represents the adjacency matrix for each subgraph. The targeted links are given below for each matrix. The first matrix represents subgraph for target link AB and having a common neighbor $F$. The second matrix represents a subgraph for target link BF and having a common neighbor $A$. The third matrix represents subgraph for target link AE and having a common neighbor $C$. AB, BF adjacency matrices are having same nodes in subgraph. Targeted links are denoted with "*****" symbol in the matrices. CNN will be feed by these adjacency matrices sequentially to train "Positive class". There is a big drawback when CNN is feed without node labeling. Target links are in different places in all three matrices, and the position of common neighbors are not consistent. Machine learning models cannot be trained if the order of features is inconsistent. This problem has high impact when the subgraph size is bigger. PLACN proposed a new algorithm for node labeling in order to solve the inconsistency problem in adjacency matrices over data points.

PLACN proposes an algorithm which keeps target link in the same position in all adjacency matrices and orders common neighbors based on average distance from target link and average weight from source and target nodes. The proposed algorithm keeps the source and target nodes in the first and second positions in the adjacency matrix so that the target link will always in the position [0,1] and [1,0]. Common neighbors are prioritized first by their average hop from the target link. Common neighbors which have the same average hops will be sorted by their average weights divided by average hop. The average distance is calculated by the equation below.

$$h_{avg}^{v} = \frac{1}{2}(h_i + h_j) \tag{3.4}$$

Where $h_i$ is the shortest distance from node $i$ to common neighbor node $v$ and $h_j$ is shorted distance from node $j$ to common neighbor node $v$. Then distances are calculated in hops and averaged to get distance from link $ij$. Another score is calculated

for nodes in the subgraph to sort them when the same average distance common neighbors are existing. The average weight can be calculated by the equation below.

$$w_{avg}^{v} = \frac{1}{2}\left(\frac{1}{h_i}\sum_{p=0}^{h_i} w_p^i + \frac{1}{h_j}\sum_{p=0}^{h_j} w_p^j\right) \tag{3.5}$$

Where $\sum_{p=0}^{h_i} w_p^i$ is the total weight in the path from common node $v$ to node $i$. $\sum_{p=0}^{h_j} w_p^j$ is the total weight in the path from common node $v$ to node $j$. Each path weights are divided by their corresponding hops to get average path weight. Finally, the average of both path weight is calculated and assigned to the node to sort among nodes that are having the same average hop from the link.

**Algorithm 2** Subgraph Labeling

**Input**: Nodes List $N_K$, Target link $E_{ij}$, Subgraph $\langle S \rangle$
**Output**: Ordered nodes list $O_K$

$O_K = \{i, j\}$
$R_K = N_K - \{i, j\}$
$M\langle k : \langle w_{avg}, h_{avg}\rangle\rangle \leftarrow$ Map for node information
**for all** $v \in R_K$ **do**
    $h_i = min(dist(v, i))$
    $h_j = min(dist(v, j))$
    $w_{avg}^{v} = \frac{1}{2}\left(\frac{1}{h_i}\sum_{p=0}^{h_i} w_p^i + \frac{1}{h_j}\sum_{p=0}^{h_j} w_p^j\right)$
    $h_{avg}^{v} = \frac{1}{2}(h_i + h_j)$
    $M \leftarrow (v : (1/w_{avg}^{v}, h_{avg}^{v}))$
**end for**
sort the map based on $h_{avg}$
        sort the map based on $w_{avg}$ for same $h_{avg}$
**for** $v$ in $M$ **do**
    $O_K \leftarrow O_K \cup v$
**end for**
**if** $|O_K| > K$ **then**
    **while** $|O_K| = K$ **do**
        remove nodes from bottom
    **end while**
**end if**
**return** $\langle O_K \rangle$

The proposed node labeling algorithm is given in algorithm 2. The algorithm takes Nodes List $N_K$, Target link $E_{ij}$, Subgraph $\langle S \rangle$ as input and returns Ordered nodes list $O_K$. The algorithm is not only orders the nodes to keep consistency but also if the given node list has nodes more than subgraph factor $K$, it removes the node from the bottom so that least prioritized nodes will be removed. Result $O_K$ will have exact number node as subgraph factor $K$.

Network given in Figure 3.4 used to illustrate a simple network with positive links and their corresponding subgraph represented as adjacency matrix in Figure 3.5. It shows how inconsistency emerged when node ordering is not considered. The proposed node labeling algorithm solves this problem and after applied the algorithm 2 to the same subgraphs presented in Figure 3.5 is showed in Figure 3.6

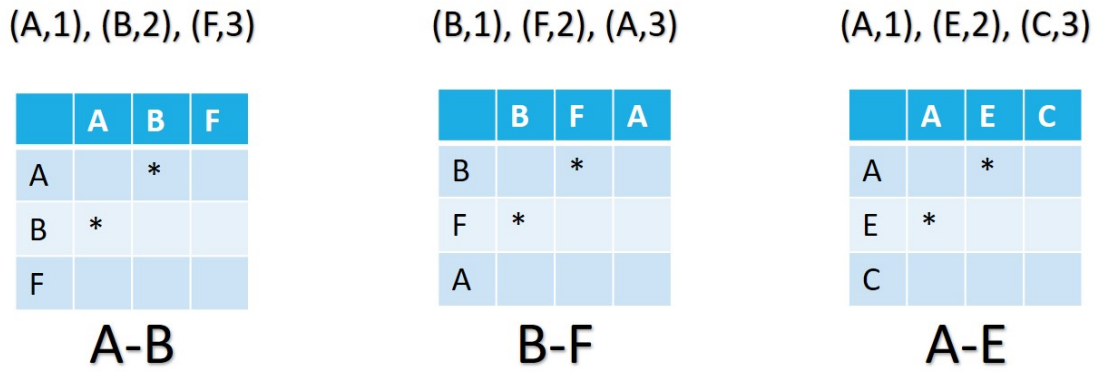


Figure 3.6: Adjacency matrices after applied proposed Node Labeling Algorithm

In these matrices, source and target nodes are getting labels 1 and 2 so that $A, B$ in the first matrix, $B, F$ in the second matrix, $A, E$ in the third matrix will get first and the second position in matrices. Common neighbors $F, A, C$ received label 3, 3, 3 in the adjacency matrix respectively. The targeted link is denoted with a symbol "*" in each matrix. After ordering nodes, the target link is in positions [0,1] and [1,0] in all matrices. Common neighbors are also in the same order in overall matrices.

Order of features are consistent in overall data points after node labeling algorithm has been applied.

## 3.5 Constructing Adjacency Matrices

PLACN tends to improve the accuracy of link prediction by analyzing common neighbors between targeted nodes. In order to analyze nodes and their relationship with target link, we need to calculate the features of common nodes. Heuristics scores are similarity score which represents a similarity value between two given nodes. Extracted subgraph will contain information on how they interconnected with each other, and similarity scores will give information on how strong they relate to each other. Past researches calculated different heuristic scores between only targeted nodes and try to predict links based on the score. Even though some heuristic score performs well in certain types of social networks not performed well in other types. A combination of scores performed better than a single heuristic score. No past research analyzed common neighbors and how they influence the targeted node for the formation of links. PLACN considers five different heuristics, and they are given below.

- Common Neighbors $|\Gamma(i) \cap \Gamma(j)|$
- Jaccard Coefficient $\frac{|\Gamma(i) \cap \Gamma(j)|}{|\Gamma(i) \cup \Gamma(j)|}$
- Adamic-Adar $\sum_{k \in |\Gamma(i) \cup \Gamma(j)|} \frac{1}{log|\Gamma(k)|}$
- Preferential Attachment $|\Gamma(i).\Gamma(j)|$
- Resource Allocation $\sum_{k \in |\Gamma(i) \cup \Gamma(j)|} \frac{1}{|\Gamma(k)|}$

Common Neighbors and Preferential Attachment are first order heuristics functions and others are second order heuristics functions.

### 3.5.1 Common Neighbors

Common neighbors score is a similarity score based on a number of common neighbors between targeted nodes [28]. The formula for finding common neighbors is given below.

$$|\Gamma(i) \cap \Gamma(j)| \tag{3.6}$$

Where $\Gamma(i)$ is the set of nodes adjacent to node $i$, and $\Gamma(j)$ is the set of nodes adjacent to node $j$. A value of 0 indicates that two nodes are not sharing any common nodes, while higher values indicate nodes are closer.

### 3.5.2 Adamic Adar Index

The Adamic Adar algorithm was introduced in 2003 by Lada Adamic and Eytan Adar to predict links in a social network[2]. It is computed using the following formula:

$$\sum_{k \in |\Gamma(i) \cup \Gamma(j)|} \frac{1}{log|\Gamma(k)|} \tag{3.7}$$

Where $\Gamma(i)$ is the set of nodes adjacent to node $i$, $\Gamma(j)$ is the set of nodes adjacent to node $j$, and $\Gamma(k)$ is the set of nodes adjacent to node $k$. The score is calculated based on the node degree of common neighbors between targeted nodes.

### 3.5.3 Jaccard Coefficient

A similarity metric that is commonly used in information retrieval[2]. This score measures the probability that both x and y have a feature f, for a randomly selected feature f that either x or y has. Features are neighbors here in networks. It is calculated by dividing the number of common neighbors by total neighbors shared.It is computed using the following formula:

$$\frac{|\Gamma(i) \cap \Gamma(j)|}{|\Gamma(i) \cup \Gamma(j)|} \tag{3.8}$$

Where $\Gamma(i)$ is the set of nodes adjacent to node $i$, and $\Gamma(j)$ is the set of nodes adjacent to node $j$. A value of 0 indicates that two nodes are not sharing any common nodes, while higher values indicate nodes are closer.

### 3.5.4 Preferential Attachment

Preferential attachment means that the more connected a node is, the more likely it is to receive new links. Albert-László Barabási and Réka Albert popularised this algorithm through their work on scale-free networks[8]. It is computed using the following formula:

$$|\Gamma(i).\Gamma(j)| \tag{3.9}$$

Where $\Gamma(i)$ is the set of nodes adjacent to node $i$, $\Gamma(j)$ is the set of nodes adjacent to node $j$.A value of 0 indicates that two nodes are not close, while higher values indicate that nodes are closer.

### 3.5.5 Resource Allocation

The Resource Allocation algorithm was introduced in 2009 by Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang as part of a study to predict links in various networks[44]. It is computed using the following formula:

$$\sum_{k \in |\Gamma(i) \cup \Gamma(j)|} \frac{1}{|\Gamma(k)|} \tag{3.10}$$

Where $\Gamma(i)$ is the set of nodes adjacent to node $i$, $\Gamma(j)$ is the set of nodes adjacent to node $j$, and $\Gamma(k)$ is the set of nodes adjacent to node $k$.A value of 0 indicates that two nodes are not close, while higher values indicate nodes are closer.

The heuristics features that are mentioned above are calculated for all nodes in the

subgraph. In addition to that, weights are considered another feature in PLACN to get link information among nodes in the extracted subgraph. Each heuristic feature is constructed in an adjacency matrix, and six adjacency matrices will be stacked for a single link. The result feature matrix will have a size of $K \times K \times 6$ for each link.

Feature matrix with weights will have information about link exitance between targeted nodes. This will lead to creating a bias in CNN training. CNN will learn a pattern that if there are any non zero values in weight adjacency matrix at position [0,1] and [1,0], then it will be positive and if the values are zero in weight adjacency matrix at position [0,1] and [1,0], then it will be negative. Since testing links always have zero values at the given position above, CNN will always classify them as negative class. To overcome this problem, we put value 0 at location [0,1] and [1,0] in the weight feature matrix. Figure 3.7 illustrates that a positive link with feature matrix and in weight matrix position [0,1] and [1,0] are containing the value 0.

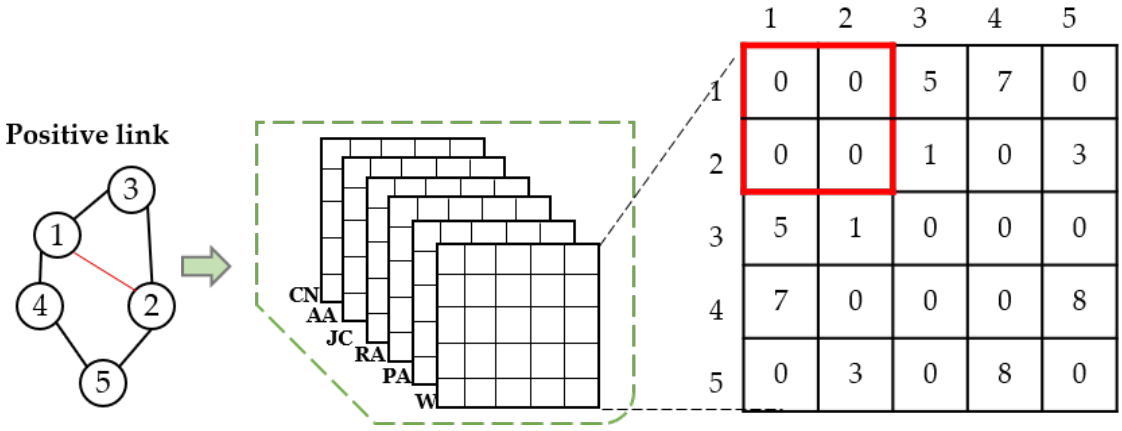


Figure 3.7: Adjacency matrices of a subgraph and enlarged weight matrix

Feature matrices are symmetric, and diagonal values are 0 since they represent nodes with a connection themselves. CNN is used as a classifier in PLACN to learn a pattern. Neural Networks have many parameters, and they are optimized during

the training phase. Neural Networks take numeric values as input and try to adjust weights with class. If input values are in a vast range, then optimizing weight becomes hard and will not get a global minimum. Feature matrices with different similarity scores will have different ranges. This creates optimizing hard for Neural Network. We used normalization to keep all values between rage 0 and 1. Normalization can be achieved using the following equation.

$$x = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{3.11}$$

Firstly, all feature matrices are stored as a dataset with real values. Then normalization is performed for each layer /feature matrix. For example, only the Adamic Adar feature matrix will be extracted for all data points, and normalization is performed so that only Adamic Adar minimum and maximum will affect during normalization. Like the example, normalization will be performed for all layers individually. Finally, all normalized data points will be used to train CNN.

## 3.6 Convolutional Neural Network

PLACN uses a CNN to classify the links as positive and negative. A positive link represents that link will occur in the near future, and a negative link represents it will not. PLACN converts the link prediction problem into an image classification problem by creating a feature matrix with six different heuristics. Feature matrix can be compared with images color images usually have three channels, which are RGB (RED, GREEN, BLUE), so images are treated as a three-dimensional matrix. In PLACN, subgraphs are converted into six adjacency matrix with six heuristics. These adjacency matrices can be considered as $K \times K$ image with 6 channels. A randomly picked data point from 2 different normalized datasets are plotted in Figure 3.8

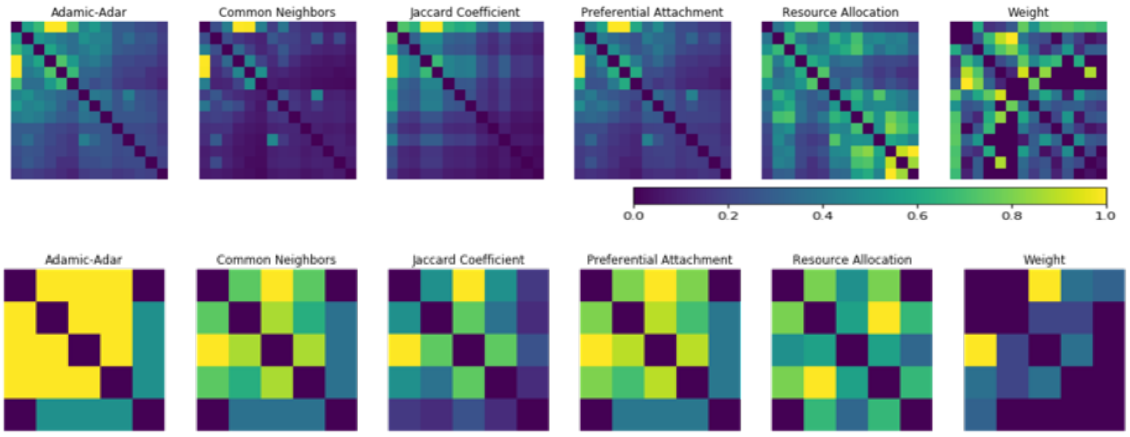


Figure 3.8: Randomly selected data points of two different datasets .Adjacency matrices are plotted

We train CNN to classify two classes, such as positive link and negative link. When training a Neural Network, the optimization is achieved by backpropagation with calculating the loss function. The loss function varies with the problem. Since this is a binary classification problem, we select Binary Cross-Entropy as the loss function for PLACN's CNN. Binary Cross Entropy is calculated by the equation as follows

$$BCE = -\sum_{i=1}^{c'=2} t_i log(f(s_i))$$
$$= -t_i log(f(s_1)) - (1 - t_i) log(f(s_1))$$

where $c'$ is the class, $t$ is the predicted label and f(s) is the predicted probability.

# Chapter 4

# Experimental Setup

In this chapter, we explain the details of the experimental setup and environments such as tools and libraries used to implement PLACN framework, systems configurations of data pre-processing step and CNN training and testing, dataset details, the detailed configuration of CNN and layer details, train and test details and evaluation method to test the results.

## 4.1 Tools and libraries

We implemented PLACN framework using Python 3.6 programming language. The libraries and their versions that are used to implement PLACN are listed below

- NetworkX 2.3
- NumPy 1.17.1
- SciPy 1.3.1
- Pandas 0.25.1
- Tensorflow GPU 1.14.0

We used PyCharm IDE, Jupyter Notebook to implement and test PLACN framework. We used two different systems to pre-process datasets and train/test CNN for PLACN.

## 4.2 System Configurations

Data pre-processing and construction of trainable dataset was created using a DELL cluster with the specification of POWER8 52 processor, 256 GB of RAM. We trained and tested PLACN's CNN in an Nvidia GTX 1050Ti, 4 GB GPU with 768 CUDA cores.

## 4.3 Datasets

The proposed PLACN framework aims to adapt to various types of social networks and improves the accuracy of link prediction. We selected seven different types of real-world social network datasets and tested them with other state-of-the-art methods. Tested datasets, and their details are given in table 4.1.

- USAir is an airline traffic network data. In this dataset, nodes represent the airports, and edges represent the air route that exists [9].
- NS is a co-authorship network dataset that has information regarding network science researchers and their collaboration[29].
- PB is a US political web-blogger relation dataset that has data about USA political web bloggers and their interaction among them [1].
- Yeast is a protein to protein interaction network which have information about the interaction between proteins of yeast[37].

| **Dataset** | $\|V\|$ | $\|E\|$ | **Ave node Degree** |
|---|---|---|---|
| USAir | 332 | 2126 | 12.81 |
| NS | 1589 | 2742 | 3.45 |
| PB | 1222 | 16714 | 27.36 |
| Yeast | 2375 | 11693 | 9.85 |
| C.ele | 297 | 2148 | 14.46 |
| Power | 4941 | 6594 | 2.67 |
| Router | 5022 | 6258 | 2.49 |

Table 4.1: The statistical information of each real-world networks.

- C.elegans is a neural network dataset that has information regarding connectivity between neurons in a C.elegans worm[39].
- Power is a power grid dataset which contains information of the western US electric grids and their connectivity[39]
- Router is an internet routing dataset which has information about in which route data transfer among servers and their connections [35]

## 4.4 CNN configurations

PLACN has a CNN as a classifier to predict positive links and negative links. CNN is configured with different layers and activation functions. Details of CNN layers and tunable parameters are given in the table 4.2.

We used 80% data to train, 10% data to validate, and 10% test the model. We used the same train test split ratio for other state-of-the-art methods. We trained our model for 50 epochs. We used 10% validation data to test during training and saved the best model with the lowerest validation loss score. After training, we loaded the

| Layers | Specification | Activation func | No. parameters |
|---|---|---|---|
| Convolutional | 32 filters with $4 \times 4$ size | relu | 3104 |
| Average Pooling | $5 \times 5$ kernal size | N/A | N/A |
| Flatten | Flattens pooled matrix | N/A | N/A |
| Dense layer | 300 neurons | relu | 240300 |
| Dense layer | 128 neurons | relu | 38528 |
| Output | 1 neuron | sigmoid | 129 |

Table 4.2: PLACN's Convolution Neural Network configurations .

best-saved model tested the data.

## 4.5 Training and Testing

PLACN takes network data and subgraph factor $K$ as input. Subgraph factor K is calculated for all datasets then datasets, and their corresponding K is given to PLACN's preprocessing component in the IBM cluster. Social network datasets are not having the same amount of positive links and negative links. Usually, links that are not present always very higher than the links exist. Machine learning models are trained with an equal number of data points in each class to learn a generalized model. If a class has a higher number of data points than others, then the model will be biased to the class with higher data points.

We overcome the class imbalance problem with undersampling technique. We extracted subgraphs for all links that are present in the given dataset and stored it as positive class data. We randomly picked the same number of links that are not present in the given network dataset and stored it as negative class data. Now positive and

negative class have the same number of data points, that leads to unbiased training.

## 4.6 Evaluation metric

In machine learning, the evaluation of a model is an essential task in order to measure its performance. Usually, accuracy is taken place as a performance metric. However, accuracy is not good enough to measure the performance of a model if it is a binary classification problem. In a two-class problem, a model can make predictions randomly and still have a chance that its predictions may be correct. In such a case, we need another metric that can evaluate the model, whether it is predicting randomly or learned to separate classes. We select Area Under a Curve - Receiver Operating Characteristics (AUC -ROC) as the evaluation metric. Details of AUC-ROC and its calculation methodology are given in subsection.

### 4.6.1 AUC -ROC

AUC - ROC is an evaluation metric that widely used in binary classification problems. AUC-ROC measures the prediction model in different threshold settings. ROC is a curve that plots probabilities between True Positive Rate and False Positive Rate in different threshold values. AUC tells the degree of the separability of the given model. AUC value is in the range from 0 to 1. AUC reaches one, represents that the given model can predict positive class as a positive class and negative class as a negative class. Higher AUC presents that the given model is not making predictions randomly, and trained in a way that learned a pattern to separate classes. Calculation of AUC-ROC and explanation of steps are given in the following.

PLACN treats link prediction as a binary classification problem by categorizing links in two classes, which are the positive link class and negative link class. Positive

class depicts that the given link will exist in the near future, and the negative class depicts that the given link will not exist in the near future. PLACN model is trained to make a prediction that positive class as a positive and negative class as a negative. Based on predicted labels and actual class, we can construct a confusion matrix.

| | | Actual | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted | Positive | True Positive | False Positive |
| | Negative | False Negative | True Negative |

Figure 4.1: Confusion Matrix

Where True Positive (TP) is given as the number of data points that model predicted as a "positive" class, which are having an actual label "positive." True Negative (TN) is given as the number of data points that the model predicted as a "negative" class, which are having an actual label "negative." False Positive (FP) is given as the number of data points that the model predicted wrongly as a "positive" class, which are having an actual label "negative." False Negative (FN) is given as the number of data points that the model predicted wrongly as a "negative" class, which are having an actual label "positive."

Our model will predict probabilities for each data point. Defining a threshold for the predicted probabilities will classify data points. Different thresholds will make the prediction fall into a certain class. At different thresholds, we can calculate TP, TN, TN, FN, and make confusion matrices. We need to calculate the ratio of correct classification and false classification.

True Postive Ratio (TPR) is a ratio that represents the proposed model's ability to capture correct classification. TPR is also referred with the terms "Specificity" and "Recall". TPR can be calculated from a given confusion matrix by the following equation.

$$TPR(TruePositiveRate) = \frac{TruePositive}{TruePositive + FalseNegative} \tag{4.1}$$

False Postive Ratio (FPR) is a ratio that represents the proposed model's incorrect classification rate. FPR can be calculated from a given confusion matrix by the following equation.

$$FPR(FalsePositiveRate) = \frac{FalsePositive}{TrueNagative + FalsePositive} \tag{4.2}$$

ROC curve is used to plot the ratio between TPR and FPR in different thresholds. The curve is plotted with TPR is on the y-axis, and FPR is on the x-axis. The linear line of the ROC curve represents that the classifier is making random predictions and curve approaching the highest TPR rate, and the lower FPR rate is a better classifier. An example ROC curve is illustrated in Figure 4.2.

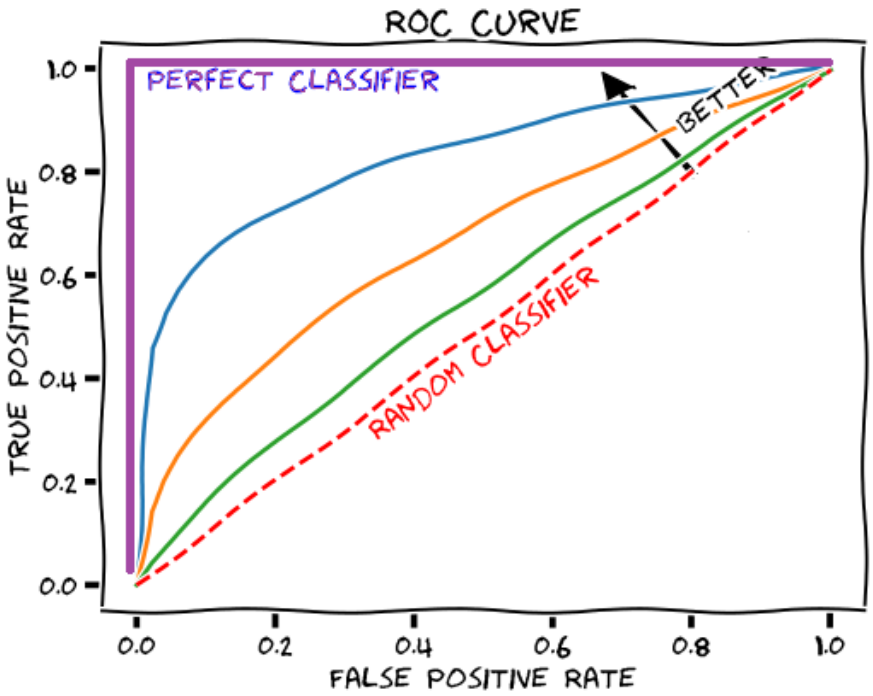


Figure 4.2: ROC curve

In the illustrated graph, the purple line indicates a perfect classifier behavior, and the dashed red line shows the behavior of a random predictor. Area Under Curve (AUC) is a perfect way to measure of a classifier, whether it is a random predictor or have the ability to classify data. A higher AUC score represents the tested model is a better classifier.

We compared the performance of PLACN in seven datasets with heuristics scores given in section 3.1, SEAL, WLNM methods, and four state-of-the-art latent feature methods. Experimental settings for methods that we compared are listed below.

- WLNM: we used the implemented model available at

  https://github.com/muhanzhang/LinkPrediction and set hop count as 3 (same setting as in their paper) to test all seven datasets.

- SEAL: we used the implemented model available at

  https://github.com/muhanzhang/SEAL and set hop count as 3 (same setting as in their paper) to test all seven datasets.

- SBM: we used the implementation available at

  http://tuvalu.santafe.edu/ aaronc/wsbm/ with a setting of a latent group number 12

- Node2Vec: we used the implementation available at

  https://github.com/eliorc/node2vec and used the default settings that come with the software.

- LINE: we used the implementation available at

  https://github.com/tangjianpku/LINE and used the default settings that come with the software.

- VGAE: we used the implementation available at

  https://github.com/tkipf/gae and used the default settings that come with the software.

We ran the test in each method ten times and calculated mean AUC score and standard deviation of AUC scores and compared them with the proposed PLACN. Results and comparisons of them are presented, discussed, and analyzed in the next section.

# Chapter 5

# Discussions, Comparisons and Analysis

In this chapter, we present the results of the conducted various tests. We recorded PLACN's performance against first-order heuristics methods Common Neighbors, Jaccard Coefficient and Preferential Attachment, second-order heuristics Adamic-Adar and Resource Allocation, high-order heuristics Katz and Page Rank, latent methods LINE, SBM, Node2Vec and VGAE, and lastly we compare with state-of-the-art methods WLNM and SEAL. We conducted our experiments on seven different types of social network datasets with the experimental setup explained in Chapter 4.

We illustrate the performance of all compared methods on each dataset as bar charts. Charts plots average AUC score of all methods on 10 test runs on each dataset. We further discussed the performance of the PLACN's with proposed subgraph factor $K$ and varying subgraph size. Finally, analyze the computational complexity of the proposed PLACN framework against the state-of-the-art method.

## 5.1 Comparison and Analysis

### 5.1.1 Performance comparison on Network Scientists dataset

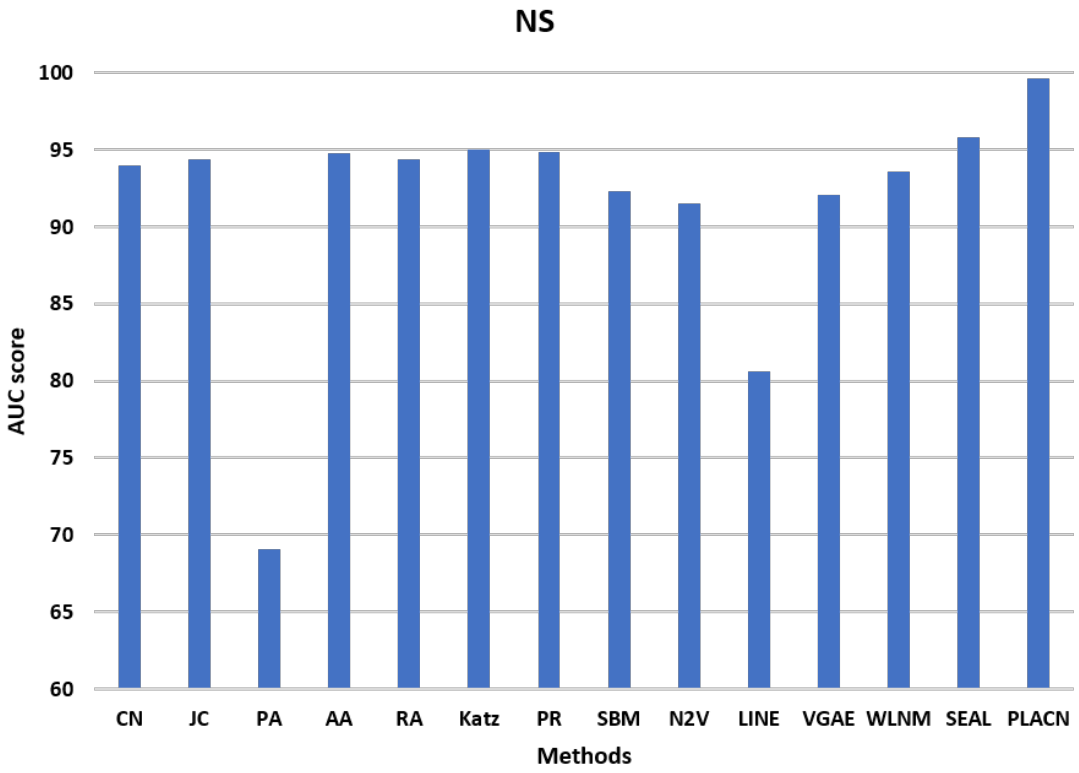


Figure 5.1: AUC scores of other methods that are tested on NS dataset compared with PLACN's AUC score

NS is a co-authorship dataset with 1589 nodes and 2742 edges. Heuristic scores such as Common Neighbors (CN), Jaccard Coefficient (JC), Preferential Attachment (PA), Resource Allocation (RA), Katz, Page Rank (PR) are compared with PLACN. Even though PA not performing well, other heuristics are showing similar results. Latent method Large-scale Information Network Embedding (LINE) is showing that the method is not suitable for co-authorship type social networks. Other latent methods are showing lower performance than most of the heuristic methods. State-of-the-art subgraphing methods WLNM and SEAL are showing similar results as most heuristics. PLACN outperforms all methods compared and showing significant increment

than latent and heuristics methods.

### 5.1.2 Performance comparison on USAir dataset

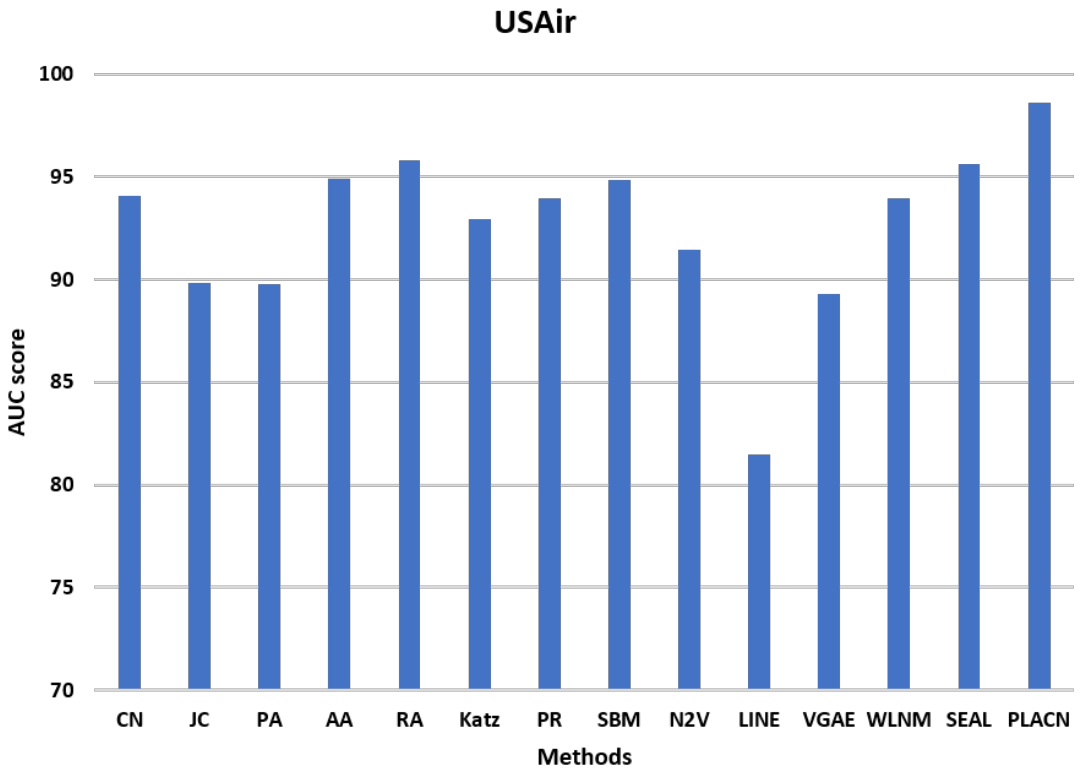


Figure 5.2: AUC scores of other methods that are tested on USAir dataset compared with PLACN's AUC score

USAir is an air traffic dataset with 332 nodes and 2126 edges. Heuristic scores such as Common Neighbors (CN), Jaccard Coefficient (JC), Preferential Attachment (PA), Resource Allocation (RA), Katz, Page Rank (PR) are compared with PLACN. Even though PA and JC not performing well, other heuristics are showing similar results. While SBM is showing similar results as best heuristic AA, Latent method Large-scale Information Network Embedding (LINE) is showing that the method is not suitable for air traffic type social networks. Other latent methods are showing lower performance than most of the heuristic methods.state-of-the-art subgraphing

methods WLNM and SEAL are showing similar results as most heuristics. PLACN outperforms all methods compared and showing significant increment than latent, heuristics, and subgraphing methods.

### 5.1.3 Performance comparison on Political Bloggers dataset

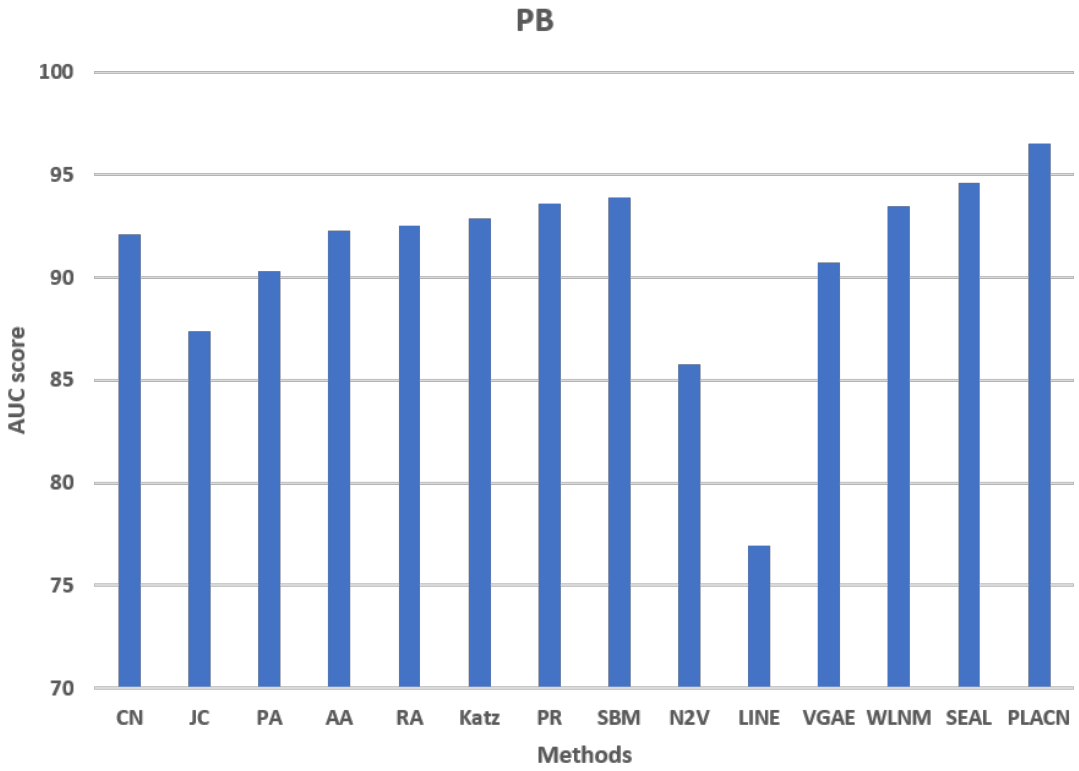


Figure 5.3: AUC scores of other methods that are tested on PB dataset compared with PLACN's AUC score

PB is a USA political blogger's connectivity dataset with 1222 nodes and 16714 edges. Heuristic scores such as Common Neighbors (CN), Jaccard Coefficient (JC), Preferential Attachment (PA), Resource Allocation (RA), Katz, Page Rank (PR) are compared with PLACN. Even though PA and JC not performing well, other heuristics are showing similar results. While SBM is showing similar results as best heuristic PR, Latent method Large-scale Information Network Embedding (LINE) is showing

that the method is not suitable for bloggers connectivity type social networks. Other latent methods are showing lower performance than most of the heuristic methods. State-of-the-art subgraphing methods WLNM and SEAL are showing similar results as most best heuristics. PLACN outperforms all methods compared and showing significant increment than latent, heuristics, and subgraphing methods.

### 5.1.4 Performance comparison on Yeast dataset

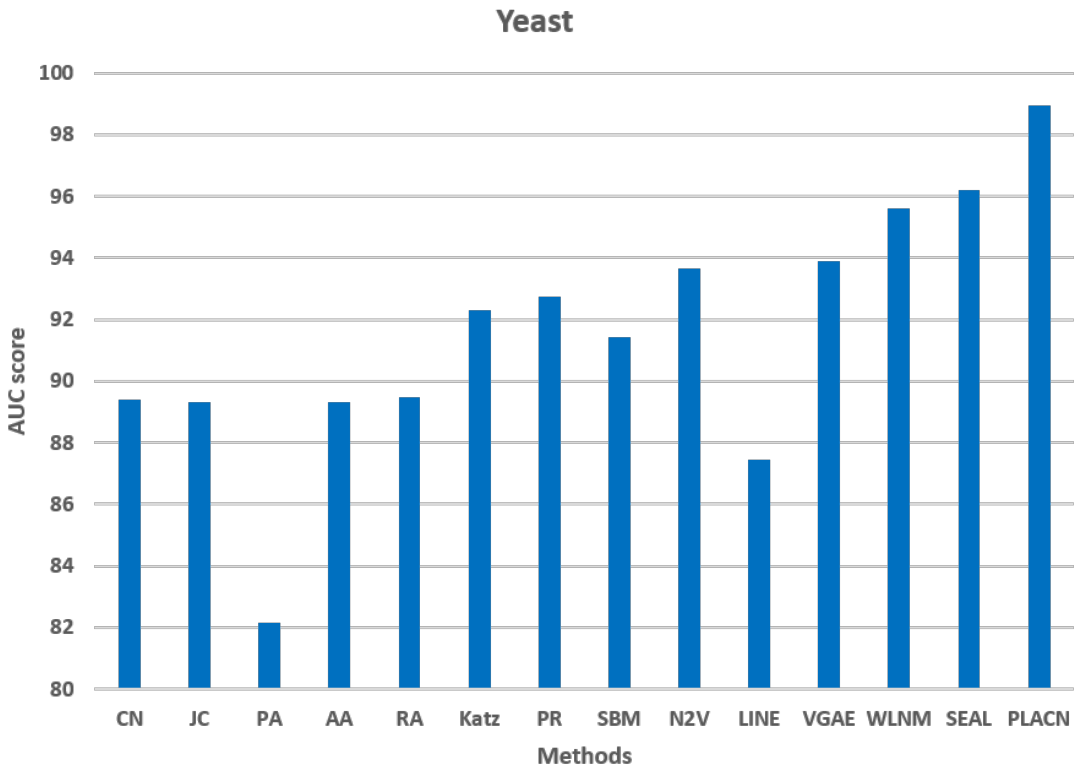


Figure 5.4: AUC scores of other methods that are tested on Yeast dataset compared with PLACN's AUC score

Yeast is a particular type of yeast's protein-to-protein interaction network dataset with 2375 nodes and 11693 edges. Heuristic scores such as Common Neighbors (CN), Jaccard Coefficient (JC), Preferential Attachment (PA), Resource Allocation (RA), Katz, Page Rank (PR) are compared with PLACN. Even though PA is not perform-

ing well, other low order heuristics are showing similar results. While high order heuristics Katz and PR are showing best heuristic results, Latent method Large-scale Information Network Embedding (LINE) is showing that the method is not suitable for protein-to-protein interaction type social networks. N2V and VGAE latent methods are showing higher performance than all heuristic methods. State-of-the-art subgraphing methods WLNM and SEAL are showing better results than latent and heuristics methods. PLACN outperforms all methods compared and showing significant increment than latent, heuristics, and subgraphing methods.

### 5.1.5 Performance comparison on C.elegans dataset

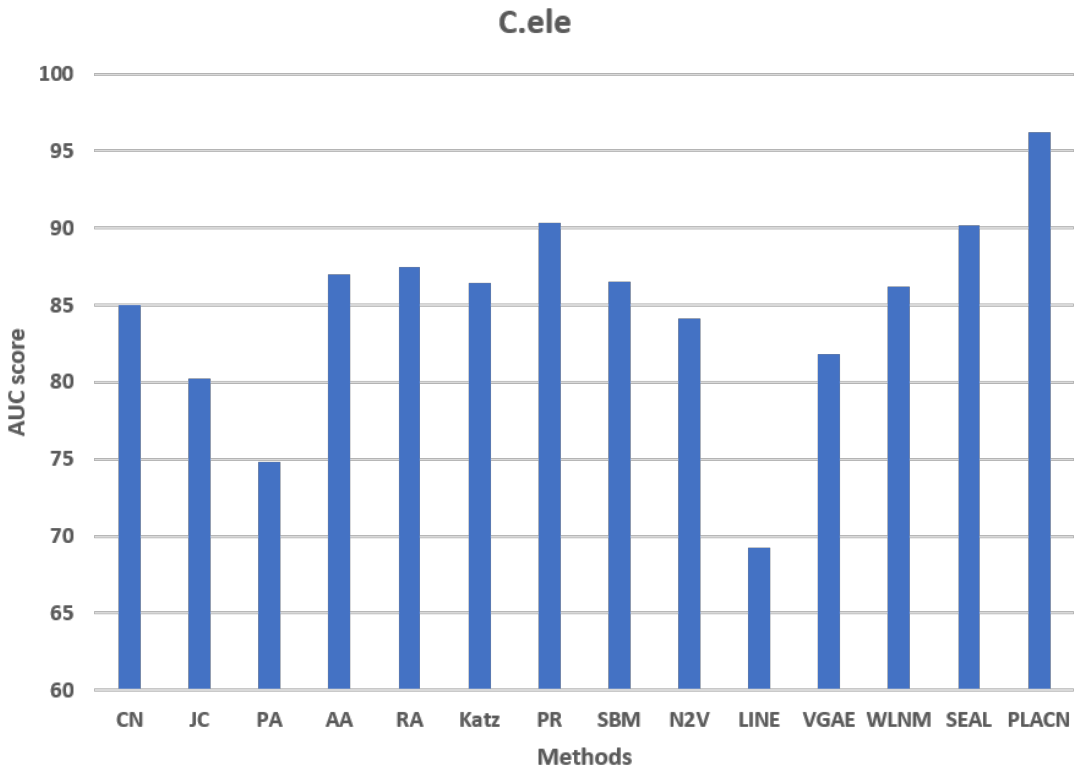


Figure 5.5: AUC scores of other methods that are tested on C.elegans dataset compared with PLACN's AUC score

C.elegans is a particular type worm and the dataset is based on its neuron interaction with 297 nodes and 2148 edges. Heuristic scores such as Common Neighbors (CN), Jaccard Coefficient (JC), Preferential Attachment (PA), Resource Allocation (RA), Katz, Page Rank (PR) are compared with PLACN. Even though PA is not performing well, AA, RA, Katz are showing similar results. While high order heuristic PR is showing best heuristic results, Latent method Large-scale Information Network Embedding (LINE) is showing that the method is not suitable for neurons interaction type social networks. All other latent methods are showing lower performance than most heuristics methods. State-of-the-art subgraphing methods WLNM and SEAL are showing similar results as best heuristics methods. PLACN outperforms all methods compared and showing significant increment than latent, heuristics, and subgraphing methods.

### 5.1.6 Performance comparison on Power Grids dataset

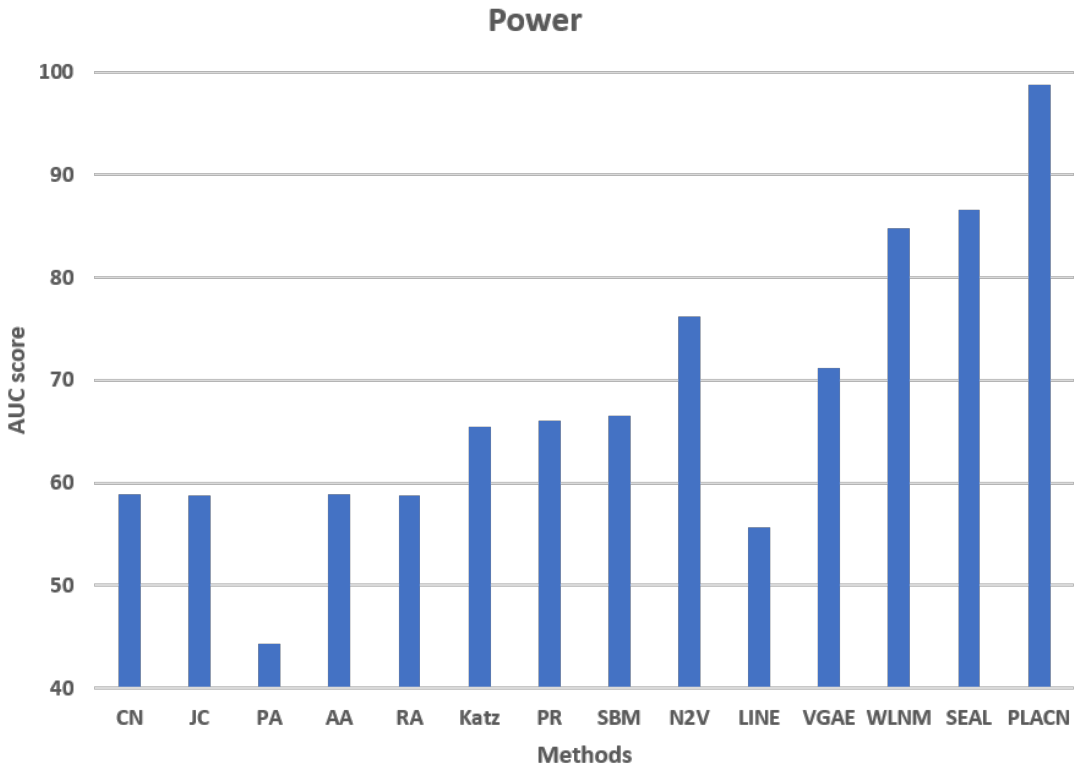


Figure 5.6: AUC scores of other methods that are tested on Power dataset compared with PLACN's AUC score

Power is USA power grid connectivity dataset with 4941 nodes and 6594 edges. Heuristic scores such as Common Neighbors (CN), Jaccard Coefficient (JC), Preferential Attachment (PA), Resource Allocation (RA), Katz, Page Rank (PR) are compared with PLACN. Even though PA is showing performing below 50% AUC, PR and Katz are showing similar results. While high order heuristic PR is showing better heuristic results, Latent method Large-scale Information Network Embedding (LINE) is showing that the method is not suitable for power grid line type social networks. All other latent methods are showing higher performance than most heuristics methods . State-of-the-art subgraphing methods WLNM and SEAL are showing similar results as best heuristics methods. PLACN outperforms all methods compared

and showing significant increment than latent, heuristics, and subgraphing methods.

### 5.1.7 Performance comparison on Network Router dataset

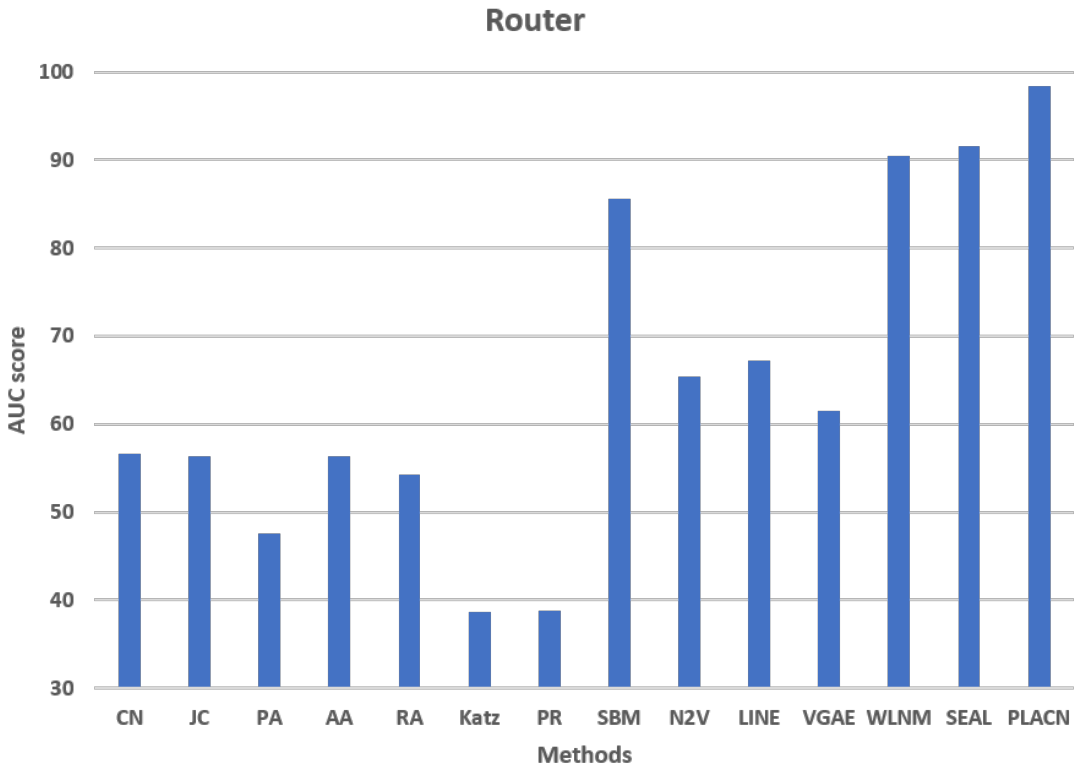


Figure 5.7: AUC scores of other methods that are tested on Router dataset compared with PLACN's AUC score

Router is data traffic dataset with 5022 nodes and 6258 edges. Heuristic scores such as Common Neighbors (CN), Jaccard Coefficient (JC), Preferential Attachment (PA), Resource Allocation (RA), Katz, Page Rank (PR) are compared with PLACN. Higher order heuristics PR, Katz are showing performing below 40% AUC, all heuristics are showing lower than 60% AUC results. Latent method SBM is showing best results among latent methods. All other latent methods are showing higher performance than most heuristics methods. State-of-the-art subgraphing methods WLNM and SEAL are showing better results than SBM.

| Dataset | CN | JC | PA | AA | RA | Katz | PR | WLNM | SEAL | PLACN |
|---|---|---|---|---|---|---|---|---|---|---|
| USAir | 94.04±1.02 | 89.82±1.49 | 89.76±1.23 | 94.91±1.28 | 95.79±0.98 | 92.92±1.34 | 93.96±1.17 | 95.95±1.10 | 96.33±0.82 | **98.36±0.24** |
| NS | 93.94±1.08 | 94.37±0.95 | 69.05±1.93 | 94.76±1.03 | 94.35±0.99 | 95.02±0.94 | 94.89±1.08 | 98.61±0.49 | 98.91±0.32 | **99.53±0.13** |
| PB | 92.08±0.31 | 87.39±0.46 | 90.28±0.37 | 92.30±0.39 | 92.51±0.26 | 92.89±0.45 | 93.60±0.38 | 93.49±0.47 | 94.70±0.34 | **96.67±0.44** |
| Yeast | 89.41±0.59 | 89.32±0.66 | 82.17±1.20 | 89.31±0.59 | 89.45±0.72 | 92.29±0.58 | 92.76±0.55 | 95.62±0.52 | 97.96±0.47 | **98.87±0.30** |
| C.ele | 84.97±1.72 | 80.19±1.47 | 74.83±1.94 | 87.01±1.38 | 87.49±1.41 | 86.42±1.71 | 90.32±1.52 | 86.18±1.72 | 90.15±0.86 | **96.08±0.26** |
| Power | 58.86±0.75 | 58.79±0.83 | 44.37±1.00 | 58.83±0.92 | 58.81±0.84 | 65.42±1.51 | 66.03±1.59 | 84.76±0.98 | 88.50±1.17 | **98.78±0.38** |
| Router | 56.58±0.41 | 56.39±0.46 | 47.62±1.51 | 56.39±0.51 | 56.39±0.51 | 38.62±1.42 | 38.82±1.31 | 94.41±0.88 | 96.45±0.96 | **98.40±0.38** |

Table 5.1: Comparison of AUC with heuristic methods, WLMN and SEAL. All values represent mean±std.

| Dataset | SBM | N2V | LINE | VGAE | PLACN |
|---|---|---|---|---|---|
| USAir | 94.85±1.14 | 91.44±1.78 | 81.47±10.71 | 89.28±1.99 | **98.36±0.24** |
| NS | 92.30±2.26 | 91.52±1.28 | 80.63±1.90 | 94.04±1.64 | **99.53±0.13** |
| PB | 93.90±0.42 | 85.79±0.78 | 76.95±2.76 | 90.70±0.53 | **96.67±0.44** |
| Yeast | 91.41±0.60 | 93.67±0.46 | 87.45±3.33 | 93.88±0.2 | **98.87±0.30** |
| C.ele | 86.48±2.60 | 84.11±1.27 | 69.21±3.14 | 81.80±2.18 | **96.08±0.26** |
| Power | 66.57±2.05 | 76.22±0.92 | 55.63±1.47 | 71.20±1.65 | **98.78±0.38** |
| Router | 85.65±1.93 | 65.46±0.86 | 67.15±2.10 | 61.51±1.22 | **98.40±0.38** |

Table 5.2: Comparison of AUC with Latent feature methods. All values represent mean±std.

PLACN outperforms all methods compared and showing significant increment than latent, heuristics, and subgraphing methods.

The average AUC scores of all compared methods on seven datasets and their standard deviations are presented in Tables 5.1, 5.2.

### 5.1.8 Subgraphing Factor K empirical test results

We proposed a new formula to calculate the subgraphing factor $K$. The proposed formula is relying on network properties such as Node Degree and Network Density. We claimed that the calculated subgraphing factor is an optimal value for the number of nodes in subgraphs to extract link information and reduces computational complexity. We tested our claim by varying the number of nodes in the subgraph and calculated AUC for all seven datasets. Increasing the number of common nodes in subgraph will increase computation cost. We expect that the performance of the proposed PLACN will converge in terms of AUC score after calculated $K$. The test was conducted up to $7K$ . Results of average AUC in each tested K value is plotted in Figure 5.8 for seven datasets.

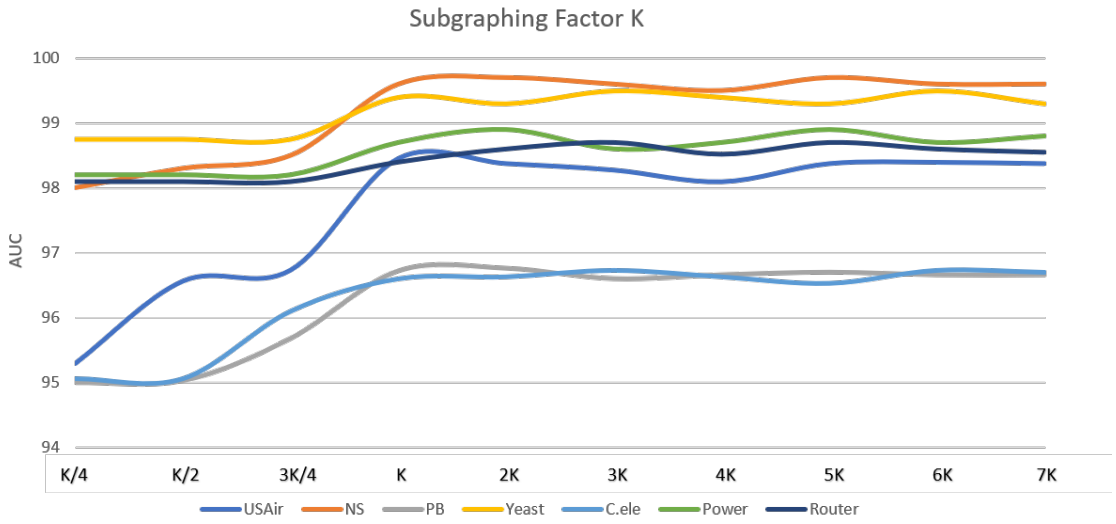


Figure 5.8: Average AUC with different subgraph sizes

Results show that after we calculated AUC at $K$, there are no significant changes in AUC. Hence the results from Graph 5.8 verifies that the proposed formula does give an optimal number for maximizing information and reducing the computational cost. We observed that after $5K$ number of nodes in the subgraph, the adjacent matrix is getting sparse on average since nodes of subgraph are tend to have lesser connections. The sparse matrix introduces a new problem that extracting features becomes harder as the adjacency matrix will contain the most values as 0. Thus, the proposed formula for calculating the subgraphing factor gives an optimal number of nodes to give information and overcomes information dilution and computational complexity problems.

### 5.1.9 Statistical significance test results

We ran our test for ten times and recorded the AUC score of the SEAL and proposed PLACN on all seven datasets. We plotted both frameworks AUC score histograms and performed a Wilcoxon test [40] to test statistical significance for the performance of PLACN. We define our hypothesis as follows.

$H_0$ *: No significant performance difference between the SEAL and PLACN models*

$H_1$ *: There is statistical significant that PLACN performs better than SEAL*

we defined $\alpha = 0.05$ and performed Wilcoxon test. We reject the Null hypothesis $H_0$ in case of p-value ¡ $\alpha$, which leads to accepting our alternative hypothesis $H_1$ that there is a statistical significant that PLACN performs better than the SEAL on the given dataset.

### 5.1.10 Statistical significance test results on USAir dataset

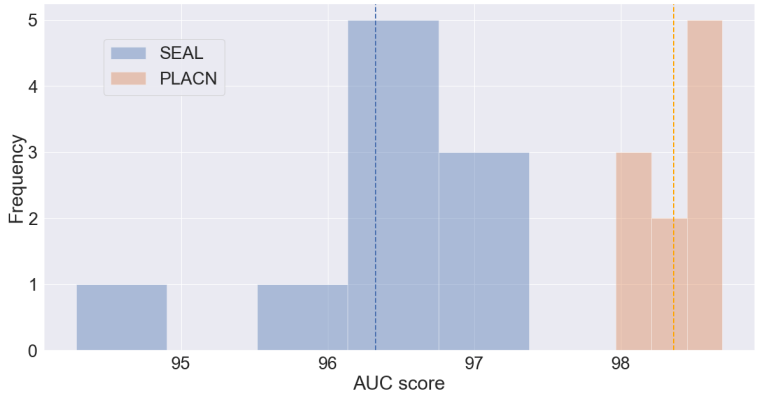


Figure 5.9: AUC distribution of SEAL and PLACN on USAir dataset

SEAL has 96.327 mean AUC and 0.82 std. PLACN has 98.356 mean AUC and 0.24 standard deviation. The Wilcoxon test was performed to test the hypothesis, which results in $P\text{-}value = 0.00506203$, and we reject $H_0$ in favour of $H_1$.

### 5.1.11 Statistical significance test results on NS dataset

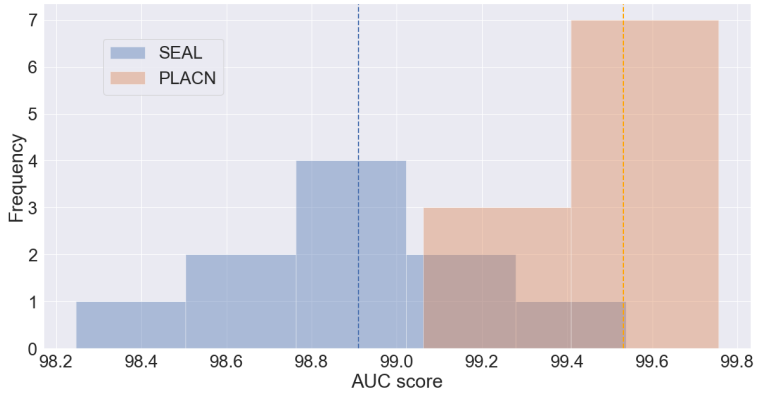


Figure 5.10: AUC distribution of SEAL and PLACN on NS dataset

SEAL has 98.909 mean AUC and 0.32 std. PLACN has 99.531 mean AUC and 0.13 standard deviation. The Wilcoxon test was performed to test the hypothesis, which results in $P\text{-}value = 0.00506203$, and we reject $H_0$ in favour of $H_1$.

### 5.1.12 Statistical significance test results on PB dataset

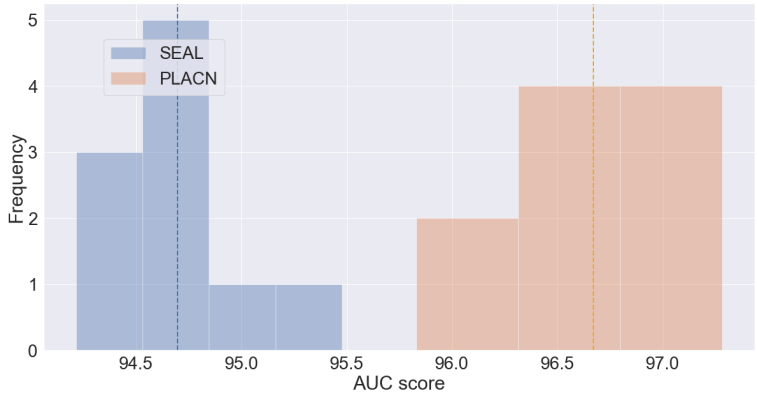


Figure 5.11: AUC distribution of SEAL and PLACN on PB dataset

SEAL has 94.696 mean AUC and 0.34 std. PLACN has 96.671 mean AUC and 0.44 standard deviation. The Wilcoxon test was performed to test the hypothesis, which results in $P\text{-}value = 0.00506203$, and we reject $H_0$ in favour of $H_1$.

### 5.1.13 Statistical significance test results on Yeast dataset

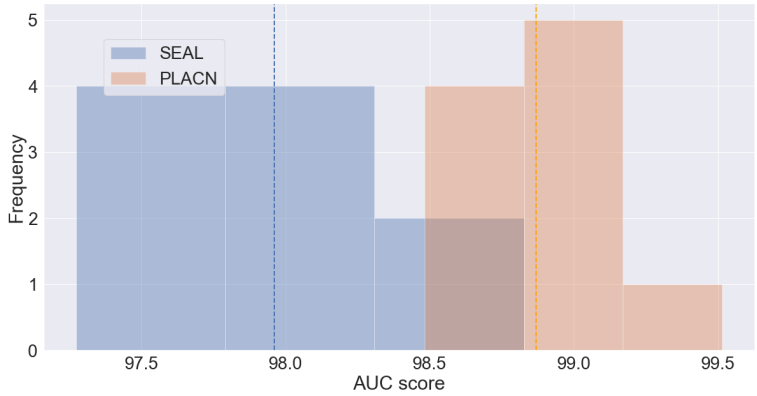


Figure 5.12: AUC distribution of SEAL and PLACN on Yeast dataset

SEAL has 97.961 mean AUC and 0.47 std. PLACN has 98.868 mean AUC and 0.30 standard deviation. The Wilcoxon test was performed to test the hypothesis, which results in $P\text{-}value = 0.00506203$, and we reject $H_0$ in favour of $H_1$.

### 5.1.14 Statistical significance test results on C.ele dataset

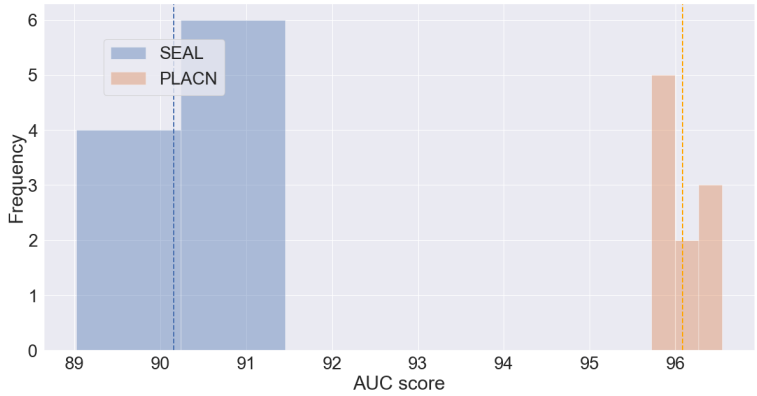


Figure 5.13: AUC distribution of SEAL and PLACN on C.elegans dataset

SEAL has 90.152 mean AUC and 0.86 std. PLACN has 96.079 mean AUC and 0.26 standard deviation. The Wilcoxon test was performed to test the hypothesis, which results in $P\text{-}value = 0.00506203$, and we reject $H_0$ in favour of $H_1$.

### 5.1.15 Statistical significance test results on Power dataset

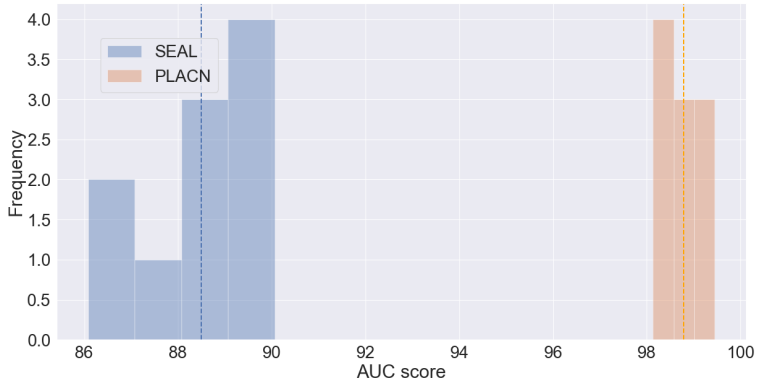


Figure 5.14: AUC distribution of SEAL and PLACN on Power dataset

SEAL has 88.499 mean AUC and 1.17 std. PLACN has 98.783 mean AUC and 0.38 standard deviation. The Wilcoxon test was performed to test the hypothesis, which results in $P\text{-}value = 0.00506203$, and we reject $H_0$ in favour of $H_1$.

### 5.1.16 Statistical significance test results on Router dataset

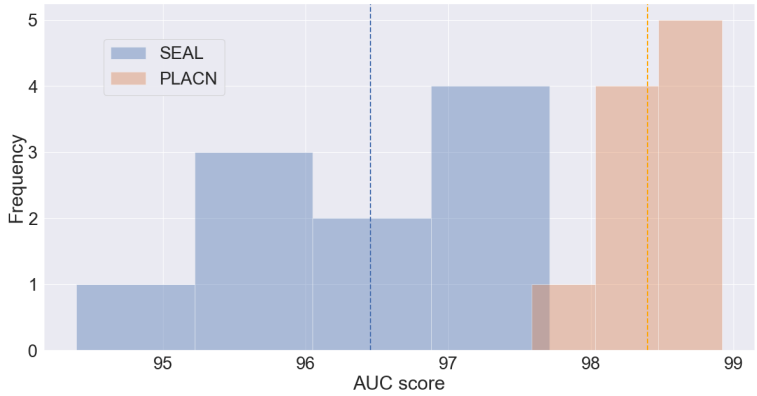


Figure 5.15: AUC distribution of SEAL and PLACN on Router dataset

SEAL has 96.451 mean AUC and 0.96 std. PLACN has 98.395 mean AUC and 0.38 standard deviation. The Wilcoxon test was performed to test the hypothesis, which results in $P\text{-}value = 0.00691043$, and we reject $H_0$ in favour of $H_1$.

### 5.1.17 Computational Complexity Analysis

PLACN includes two proposed algorithms; five heuristics score calculations and a CNN as a classifier. The first proposed algorithm for subgraph extraction. PLACN outperforms state-of-the-art method SEAL in all tested datasets. We compare our computational cost with SEAL. PLACN's proposed Subgraph extraction algorithm is having the computational cost $O(n)$, where $n$ is the number of hops, which is the same as the SEAL's subgraph extraction algorithm. Let us consider $K$ number of nodes are present in a subgraph. SEAL has the Double Radius Node Labeling algorithm for subgraph node labeling with the computational complexity of $O(K)$, which introduced inconsistency among the same order nodes. PLACN proposes a new labeling algorithm that incorporates average hop weights to label the nodes in the same order. PLACN first sort based on hop, and then sorts the nodes which are in the same order based on average hop weight from target link. The sorting introduces an additional

computational cost. In the worst case, PLACN's node labeling algorithm takes the computational complexity $O(K^2 log(K))$. SEAL calculates high order heuristic scores such as Page Rank and Katz. High order heuristics have a high computational cost, which is $O(n^3)$. PLACN does not calculate high order heuristics in subgraphs. SEAL uses a Graph Neural Network, and PLACN uses a Convolutional Neural Network.

PLACN and SEAL have the same computational complexity in the subgraph extraction method. Even though PLACN has higher computational complexity in the node labeling algorithm, it does not calculates high order heuristic scores, which reduce the computational complexity in the feature matrix construction process. Training Convolutional Neural Network does take a similar computational cost in the classification step as the SEAL. Hence, SEAL and PLACN have similar computational costs, but PLACN outperforms SEAL is all tested datasets. Test and analysis indicates that the proposed PLACN framework is the best framework for link prediction in most type of social networks

# Chapter 6

# Conclusion and Future Work

We proposed a novel framework, namely PLACN, to solve the Link Prediction problem over various types of social networks. Various heuristics methods have been proposed since specific heuristics work better on certain types of a social network but not others. Latent methods are proposed to improve the accuracy of the Link Prediction problem and work better over various types of social networks. Even though the latent method improved the accuracy of Link Prediction in some types of social networks, heuristics outperformed latent methods in other types of social networks. New state-of-the-art methods WLNM and SEAL proposed a new approach for Link Prediction problem by extracting subgraph around the target link to automate the selection of heuristic method for given social networks. Subgraphing methods show significant improvement in Link Prediction but have some limitations and do not perform well in some types of social networks. Therefore, there is a need for modeling a new framework to solve the limitations in the state-of-the-art method and improves the accuracy. This thesis work focuses the new framework, PLACN for the link prediction problem.

PLACN implemented over the assumptions that the given input graphs are uni-

directional weighted graphs and not having parallel edges to improve the Link Prediction problem and adapts to various types of Social Networks. The framework improves accuracy by analyzing common neighbors of each target link and learns the best combination of heuristics for the given social network. PLACN consists of two new algorithms for subgraph extraction and subgraph node labeling, respectively. The proposed node labeling algorithm solves the inconsistency of node over subgraphs. However, resolving node order inconsistency adds additional computational cost to the framework. The proposed node labeling algorithm takes hop count as well as average path weight into account to order nodes in the subgraph. This calculation increases the computational complexity of this framework. This additional complexity is a limitation of this framework, and we research further to reduce the time complexity.

The framework introduces subgraph factor $K$, and the proposed formula to calculate $K$ is derived from network properties. Empirical tests show that the calculated $K$, in fact, an optimal number for subgraph size. In some networks, the average node degree and the network density are very low that yields the subgraphing factor less than five. In such cases, we consider five as the subgraph factor. PLACN converts the Link Prediction problem into Image Classification Problem and classifies them with CNN. Experiment results represent that PLACN outperforms state-of-the-art methods in all tested datasets. We evaluated the statistical significance of PLACN's performance by conducting Wilcoxon test on the AUC score of both the state-of-the-art SEAL and PLACN in all seven datasets. The statistical test showed that there is statistical significant that PLACN performs better than SEAL.

## 6.1 Future Work

We tested our framework and various types of Social Networks. However, we would like to test our framework on large social networks and extend our work on dynamic social networks. The proposed method can be extended to solve the Temporal Link Prediction in the future. PLACN has the scope that it can be further researched to apply to improve recommender systems and knowledge graph completion.

# Bibliography

[1] Robert Ackland et al. Mapping the us political blogosphere: Are conservative bloggers more prominent? In *BlogTalk Downunder 2005 Conference, Sydney*. BlogTalk Downunder 2005 Conference, Sydney, 2005.

[2] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[3] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. *Journal of machine learning research*, 9(Sep):1981–2014, 2008.

[4] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *SDM06: workshop on link analysis, counter-terrorism and security*, 2006.

[5] Wadhah Almansoori, Shang Gao, Tamer N Jarada, Abdallah M Elsheikh, Ayman N Murshed, Jamal Jida, Reda Alhajj, and Jon Rokne. Link prediction and classification in social networks and its application in healthcare and systems biology. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 1(1-2):27–36, 2012.

[6] Soner Aydın. *Link prediction models for recommendation in academic collaboration network of Turkey*. PhD thesis, 2017.

[7] Lars Backstrom and Jure Leskovec. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 635–644. ACM, 2011.

[8] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[9] V Batagelj and A Mrvar. Pajek datasets http://vlado. fmf. uni-lj. si/pub/networks/data/mix. *USAir97. net*, 2006.

[10] Michele A Brandão, Mirella M Moro, Giseli Rabello Lopes, and José PM Oliveira. Using link semantics to recommend collaborations in academic social networks. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 833–840. ACM, 2013.

[11] Doina Bucur, Giovanni Iacca, Andrea Marcelli, Giovanni Squillero, and Alberto Tonda. Multi-objective evolutionary algorithms for influence maximization in social networks. In *European Conference on the Applications of Evolutionary Computation*, pages 221–233. Springer, 2017.

[12] Hsinchun Chen, Xin Li, and Zan Huang. Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'05)*, pages 141–142. IEEE, 2005.

[13] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[14] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050, 2014.

[15] Shiping Huang, Yong Tang, Feiyi Tang, and Jianguo Li. Link prediction based on time-varied weight in co-authorship network. In *Computer Supported Cooperative Work in Design (CSCWD), Proceedings of the 2014 IEEE 18th International Conference on*, pages 706–709. IEEE, 2014.

[16] Shantha Jayalal, Chris Hawksley, and Pearl Brereton. Website link prediction using a markov chain model based on multiple time periods. *International Journal of Web Engineering and Technology*, 3(3):271–287, 2007.

[17] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*, pages 4284–4295, 2018.

[18] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[19] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[20] Mayank Lahiri and Manuel Cebrian. The genetic algorithm as a general diffusion model for social networks. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[21] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[22] Gil Levi and Tal Hassner. Age and gender classification using convolutional neural networks. In *Proceedings of the iEEE conference on computer vision and pattern recognition workshops*, pages 34–42, 2015.

[23] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3367–3375, 2015.

[24] Hao Liao and An Zeng. Reconstructing propagation networks with temporal similarity. *Scientific reports*, 5:11404, 2015.

[25] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[26] V Valli Mayil. Web navigation path pattern prediction using first order markov model and depth first evaluation. *International Journal of Computer Applications (0975-8887)*, 45(16), 2012.

[27] Megha Mishra, Harsha Dubey, and Vishnu Kumar Mishra. Page navigation prediction based on longest common subsequence and apriori algorithm. *i-manager's Journal on Cloud Computing*, 3(3):27, 2016.

[28] Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical review E*, 64(2):025102, 2001.

[29] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.

[30] ME Paoletti, JM Haut, J Plaza, and A Plaza. A new deep convolutional neural network for fast hyperspectral image classification. *ISPRS journal of photogrammetry and remote sensing*, 145:120–147, 2018.

[31] Clara Pizzuti. Ga-net: A genetic algorithm for community detection in social networks. In *International conference on parallel problem solving from nature*, pages 1081–1090. Springer, 2008.

[32] Shaoqing Ren, Kaiming He, Ross Girshick, Xiangyu Zhang, and Jian Sun. Object detection networks on convolutional feature maps. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1476–1481, 2016.

[33] Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

[34] John Scott. *Social network analysis*. Sage, 2017.

[35] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring isp topologies with rocketfuel. In *ACM SIGCOMM Computer Communication Review*, volume 32, pages 133–145. ACM, 2002.

[36] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[37] Christian Von Mering, Roland Krause, Berend Snel, Michael Cornell, Stephen G Oliver, Stanley Fields, and Peer Bork. Comparative assessment of large-scale data sets of protein–protein interactions. *Nature*, 417(6887):399, 2002.

[38] Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Albert-Laszlo Barabasi. Human mobility, social ties, and link prediction. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1100–1108. Acm, 2011.

[39] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440, 1998.

[40] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

[41] Shuang-Hong Yang, Alexander J Smola, Bo Long, Hongyuan Zha, and Yi Chang. Friend or frenemy?: predicting signed ties in social networks. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 555–564. ACM, 2012.

[42] Muhan Zhang and Yixin Chen. Weisfeiler-lehman neural machine for link prediction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 575–583. ACM, 2017.

[43] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pages 5165–5175, 2018.

[44] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.

# Vita Auctoris

| | |
|---|---|
| NAME: | Kumaran Ragunathan |
| PLACE OF BIRTH: | Jaffna,Sri Lanka |
| EDUCATION: | Bachelor of Science (Hons) in Computer Science, University of Jaffna, Jaffna, Sri Lanka, 2016. |
| | Master of Science in Computer Science, University of Windsor, Windsor, Ontario, Canada, 2019. |