An Efficient Bet-GCN Approach for Link Prediction

Rahul Saxena[1,2], Spandan Pankaj Patil[3], Atul Kumar Verma[1], Mahipal Jadeja[1], Pranshu Vyas[1], Vikrant Bhateja[4], Jerry Chun-Wei Lin[5] *

[1] Department of Computer Science and Engineering, Malaviya National Institute of Technology Jaipur, Jaipur (India)
[2] Department of Information Technology, Manipal University Jaipur, Jaipur (India)
[3] Department of Electrical Engineering, Malaviya National Institute of Technology Jaipur, Jaipur (India)
[4] Department of Electronics Engineering, Faculty of Engineering and Technology, Veer Bahadur Singh Purvanchal University, Shahganj Road, Jaunpur-222003, Uttar Pradesh (India)
[5] Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen (Norway)

Received 5 July 2022 | Accepted 12 October 2022 | Early Access 1 February 2023

Abstract

The task of determining whether or not a link will exist between two entities, given the current position of the network, is called link prediction. The study of predicting and analyzing links between entities in a network is emerging as one of the most interesting research areas to explore. In the field of social network analysis, finding mutual friends, predicting the friendship status between two network individuals in the near future, etc., contributes significantly to a better understanding of the underlying network dynamics. The concept has many applications in biological networks, such as finding possible connections (possible interactions) between genes and predicting protein-protein interactions. Apart from these, the concept has applications in many other areas of network science. Exploration based on Graph Neural Networks (GNNs) to accomplish such tasks is another focus that is attracting a lot of attention these days. These approaches leverage the strength of the structural information of the network along with the properties of the nodes to make efficient predictions and classifications. In this work, we propose a network centrality based approach combined with Graph Convolution Networks (GCNs) to predict the connections between network nodes. We propose an idea to select training nodes for the model based on high edge betweenness centrality, which improves the prediction accuracy of the model. The study was conducted using three benchmark networks: CORA, Citeseer, and PubMed. The prediction accuracies for these networks are: 95.08%, 95.07%, and 95.3%. The performance of the model is comprehensive and comparable to the other prior art methods and studies. Moreover, the performance of the model is evaluated with 90.13% for WikiCS and 87.7% for Amazon Product network to show the generalizability of the model. The paper discusses in detail the reason for the improved predictive ability of the model both theoretically and experimentally. Our results are generalizable and our model has the potential to provide good results for link prediction tasks in any domain.

Keywords

Graph Convolution Network (GCN), Graph Theory, Link Prediction, Network Centrality, Social Networks.

DOI: 10.9781/ijimai.2023.02.001

I. Introduction

Social Networks have been the primary source of information exchange between people for more than a decade now. The flow of information in this era depends heavily on the interactions of people with their peers and friends, such as liking a post, following a page, buying products, etc. Both the social networking websites and the e-commerce website are influenced by this fact. Miao et al. [1] discusses the impact of online customer reviews on product returns. The study found that the influence is even greater for sellers with good quality or branded products. Ullal et al. [2] also concluded in their study that customer reviews can significantly influence the selling and buying behavior of e-commerce companies. There are many such studies that prove how important the connections a person has are. A person's opinion and thinking are strongly influenced by the views and activities of their social environment. This ideology, in turn, is used by companies to identify the potential customers/buyers in the near future. This is done by analyzing the network of existing customers and identifying people who have the same preferences, characteristics, etc. This correlation in the characteristics of the two people forms the basis for a friendship relationship between them. This concept of link analysis and prediction is not only useful in product recommendation, but also in various areas of network science. Link prediction in network science is an important research area to understand the growth and evolution of the network. The idea of link prediction [3], [4] and analysis is of great importance in community detection, influence analysis, anomaly

* Corresponding author.

E-mail addresses: 2019rcp9153@mnit.ac.in (R. Saxena), 2018uee1353@mnit.ac.in (S. P. Patil), 2019rcp9050@mnit.ac.in (A. K. Verma), mahipaljadeja.cse@mnit.ac.in (M. Jadeja), 2018ucp1444@mnit.ac.in (P. Vyas), bhateja.vikrant@ieee.org (V. Bhateja), jerrylin@ieee.org (J. C.-W. Lin).

detection, recommendation, etc. [5] where the available information plays an important role in identifying the linking patterns. Further, link prediction has a substantial role in the study of protein-protein interaction patterns and prediction of the linkage between the unconnected protein molecules [6]. Similarly, Marcus et al. [7] have used the link prediction to study the time-evolving criminal network. Likewise, there are many applications and related areas where link prediction has played a significant role. Although researchers have proposed various link prediction models and methods, still there is a lot of scope for improvement. With the advancements in deep learning for graphs, the task of link prediction has gained increased attention. This is because deep learning techniques for graphs provide highly accurate predictions over the limited training data.

In this paper, we present the task of link prediction using a Graph Convolutional Network (GCN). The key to this idea lies in the selection of the training pool based on network centrality. This idea is explored in detail in section 4 of the paper. As a result, the link prediction task has higher accuracy given a limited training dataset, since the aggregation of the neighborhood improves due to the selection of edges based on their importance. Therefore, the contributions of the manuscript can be highlighted as follows:

- We proposed an efficient GCN-based link prediction technique where the links of the training set are selected based on edge betweenness centrality.
- The utilized justification of edge betweenness centrality is based on the selection of the training set for GCN.
- Detailed comparison of the results obtained with the current state of the art methods for link prediction.

The flow of the paper is organized as follows: Section I gives a brief introduction to link prediction, its applicability, and the contribution of the manuscript. Section II gives an overview of the state of the art in link prediction methods. Also, Graph Convolutional Networks (GCN) and their applicability to the task of link prediction are discussed in this section. Section III discusses the proposed method, its correctness and modification of the conventional GCN-based link prediction. The section also addresses the importance of network centrality to the link prediction task. Section IV highlights the experimental setup, description of the considered datasets and explanation of the proposed model. Section V discusses the results obtained with the proposed model. In addition, the results are compared with other state-of-the-art implementations over the datasets. Finally, Section VI summarizes the results of the study and highlights some future directions to be further explored.

II. Literature Survey

This section gives a brief literature review of the state of the art, highlighting link prediction and Graph Convolutional Networks. It also discusses the latest graph deep learning based architectures and frameworks to tackle the task of *link prediction*. The section focuses on the need and scope of deep learning techniques for link prediction.

A. Link Prediction

The task of link prediction can be defined as predicting whether or not two nodes will form a link in the future.

So, given a graph, if two nodes are not connected at time t, what is the probability that they will be connected at time $(t + 1)$? Taking this idea further, there may be many unconnected nodes in the graph at a given time. So the task is to correctly predict the possible connections between nodes at a given time in the network.

To formulate this more formally, consider a graph $G(V, E)$ defined as follows:

V: Set of vertices or nodes in the graph such that

$$V = \{v_1, v_2, \dots v_n\} \ \forall\, n \geq 1$$

E: Set of edges or links as $E = \{e_1, e_2, \dots e_m\} \ \forall\, m \geq 1$

This is the graphical structure at time t_0. At some time $t_1 > t_0$ the graphical structure evolves as $G(V, E')$ suchthat $E' = \{e_1, e_2, \dots e_k\} \ \forall\, m \geq 1$ and $k \geq m$. Our goal is to predict the edge set E'' for the graph G with the same number of nodes and an increased number of edges as a result of linkages between the disconnected nodes of the graph based on the information at time t_0 of the graph. This edge set should approximate the actual edge set E'.

Fig. 1 shows a graphical network in which the dashed edges represent the possible connections between the unconnected nodes at a given time in the near future. An interesting fact about the creation of connections is that each group of nodes tries to complete its *Triadic closure* [8]. According to Granovetter's theory of *Strength of Weak Ties* [9], if there is a connection between nodes A-B and A-C, then there is a strong tendency for linkage between B-C. The statement is about the triadic closure property for graphical networks. As an extension to this, there are many node groups in the network in which a pair of nodes attempts to close triads. The links between such pairs of nodes have a high probability of appearing in the future. This is one of the main ideas behind link prediction. Another idea for predicting a link between pairs of nodes is based on the different degree of expansion of the network inside and outside the group. According to Bi et al. [10], the network expansion inside the community is high. The nodes outside the community have fewer linkages, or very few nodes are connected. Apart from these, there are several other concepts for building networks such as stochastic block model [11], stochastic block model with Bayesian context, and stochastic block model with spectral clustering [12], which is the basis for *link establishment* between nodes. Another class of concepts are measures of proximity of nodes such as common neighbors, Jaccard coefficient [13], Adamic/Adar [14], Preferential Attachment Model [15] etc., based on which link establishment between nodes can be expected. These are the conventional approaches to link prediction that have evolved over time. Various improvements to these general ideas have been developed to achieve better and more efficient results. However, the increasing size of networks, aggregation of features in the form of node attributes and information, dynamic evolution of the network, and many other factors pose challenges to the computational ease and predictive ability of the methods. Machine learning/deep learning based approaches to the problem of link prediction are therefore attracting increasing attention. Combining these general ideas with artificial intelligence (AI) and machine learning (ML) based approaches has proven to be successful. The results obtained are very accurate. The remainder of the discussion in this section therefore focuses on the current state of the art in deep learning-based approaches to link prediction over the graph.

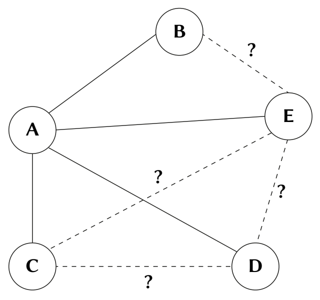


Fig. 1. Network as Graph with possible edges or links between the nodes.

B. Graph Convolutional Networks (GCN)

Graph Convolutional Networks (GCNs) have emerged in recent years as powerful machine learning methods for graph processing [16]. The basic idea behind the operation of convolutional networks is neighborhood aggregation, where the features of each node play a crucial role in decision making. Unlike an image, the structure of the graph is irregular and cannot be mapped to a fixed grid (see Fig. 2). Therefore, the structure of the graph also plays an important role. For this reason, the conventional Convolutional Neural Network (CNN) based approach cannot be used for graph structures. A GCN uses both the network structure and the features of the neighboring node to evaluate the folded value over the considered node. This additional information about the context of the node in the form of the network structure plays an important role in the prediction and classification tasks.

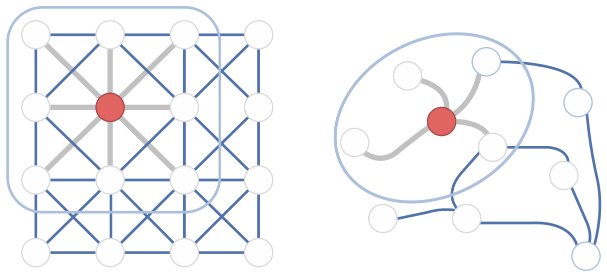


Fig. 2. Image structure v.s. graph structure [17].

The task of modeling a Graph Convolutional Network (GCN) for a graph is solved by two mathematical approaches: *Spectral Graph Theory* and *Spatial Graph Theory*. Spectral Graph Theory requires a Fourier transform based computation of translation in the frequency domain to create a graph Laplacian [17]. Since this requires a detailed mathematical explanation, we will only explain the main steps here. At a high level, the spectral graph convolution in the Fourier domain is defined by applying the filter $g\theta$ to the input signal x:

$$g\theta * x \tag{1}$$

$g\theta$: A diagonal matrix diag(θ) parameterized by $\theta \in R^n$

Since the operator based on spectral graph convolution is a position invariant of the nodes of the graph, the *graph Laplacian* matrix L for a graph G of dimensions $N \times N$ is given as:

$$L = I_N - D^{-1/2}.A.D^{-1/2} = U\Lambda U^T \tag{2}$$

Here A stands for *Adjacency Matrix*, I for *Identity Matrix*, and D for *Diagonal Matrix*. Their product gives the aggregate sum and $D^{-1/2}$ normalizes this product to suppress the effect of high degree nodes. Moreover, L can be factorized using U, which contains eigenvectors of L and Λ with the corresponding eigenvalues. Since L is a positive semi-definite matrix and U is the Fourier basis, the Fourier transform over x can be defined as follows:

$$F(x) = U^T x \tag{3}$$

Hence the inverse is presented as:

$$F^{-1}(\hat{x}) = U^T \hat{x} \tag{4}$$

If F is the Fourier domain space, the graph convolution operator can be defined as an elementwise product:

$$x * Gg = F^{-1}(F(x) \odot F(g)) = U(U^T x \odot U^T g) \tag{5}$$

Comparing equation (5) with equation (1), the final convolution equation of the graph can be given as follows:

$$x * Gg_\theta = U g_\theta U^T x \tag{6}$$

such that:

$$g_\theta = diag(U^T x), g \in R \tag{7}$$

With g_θ filled with the learning parameters $\theta^k_{i,j}$, the output on layer k can be defined as follows:

$$H^k_{:,j} = \sigma\left(\sum_{i=1}^{f_{k-1}} U\theta^k_{i,j}U^T H^{k-1}_{:,i}\right) (j = 1,2,\dots k) \tag{8}$$

Here, f_{k1-1} and f_k are the number of input and output channels in layer k, respectively, $H^k_{:,j}$ is the output channel in layer k.

However, this spectral convolution has certain limitations. First, computing the eigenvalues of the graph matrix is a computationally intensive task. Second, for very large graphs, the aggregation of neighborhoods for large values of k becomes computationally intensive and degrades the aggregation results. To solve these problems, only a neighborhood of a few hops should be considered in the localization of the filtering process. Therefore, spatial graph convolution methods have gained increasing attention. Thus, by adding formal parameters to the equation (2) and approximating the depth of the network to two, an embedding based on a 2-layer GCN model can be defined as follows:

$$Z = f(A, X) = softmax(K.ReLU(K.X.W^{(0)}).W^{(1)}) \tag{9}$$

Here K is defined as $D^{-1/2}AD^{-1/2}$. The ultimate task is to learn the weights for the model, where $C \times H$ are the trainable weights for $W^{(0)}$. Similarly, HXF are trainable weights for $W^{(1)}$. Here, C refers to the dimensions of the feature vectors, 'F' refers to the dimensions of the resulting vectors, and 'H' is the number of hidden layers. The expression in equation (9) can be further extended depending on the hidden layers in the network. The depth of the network is based on the intuition of the contribution of the k path length of the neighborhood. However, in general, graph networks do not have much impact on neighborhood interactions beyond 2 – 3 path lengths [18]. Therefore, the results of GCN networks at 2 – 3 level are remarkable and impressive; otherwise, the model suffers from the overfitting condition. The final layer of this *spatial* GCN model is guided by a *softmax* function to make predictions. The cross-entropy loss function is considered for training the model:

$$L = -\sum_{y \in Y_l}\sum_{f=1}^{F} Y_{lf} \ln Z_{lf} \tag{10}$$

Here Y_l is the set of values with their respective labels. The hyperparameters of the model are set to optimize this loss metric, including the learning rate, epochs, layer sizes, etc. A detailed discussion of these parameters can be found in section V of the paper. Further improvements to the model, such as changing the aggregation function, using weighting preferences to cluster the neighborhood, etc., provide a path to advanced versions of GCN such as Graph Attention Model, GraphSage, etc. In the following subsection, we discuss the state of the art regarding the role of GCN/GNN in efficient link prediction execution.

C. Graph Neural Network Based Approaches to Link Prediction

Since the last decade, the world has been experiencing a boom in the research area of graphical neural networks. GNN is a special kind of neural networks characterized by the structures of graphs. Semi-supervised link prediction using label propagation was first introduced by Kashima et al. [19]. This model of link prediction is applicable to multirelational domains and uses auxiliary information such as node similarity. A new fast and scalable algorithm for semi-supervised link prediction was proposed by Raymond et al. [20] for both static and dynamic graphs.

Menon et al. [21] proposed a model that predicts links through Matrix factorization. This model gains knowledge of latent features from the topological structure of the graph. Moreover, the author considered the problem of class imbalance during optimization with stochastic gradient descent and scales. Gao et al. [22] addressed the problem of predicting temporal link prediction. This model integrates the information of graph proximity, global network structure, and node content. The prediction approach called SLiPT (self-training based link prediction using a temporal network) shows better prediction accuracy and was proposed by Zeng et al. [23]. Berton et al. [24] dealt with graph construction in supervised and semi-supervised classification.

To improve performance, Kipf et al. [25] proposed VGAE (Variational Graph Auto-Encoder). This approach uses latent variables and gives better results in predicting links in citation networks. Another approach by Yang et al. [26] defines a new proximity matrix and formulates BANE (Binarized Attributed Network Embedding). In contrast to these methods, Tran et al. [27] focused on a simple but effective architecture. This architecture, named MTGAE (Multi-Task Graph Auto-Encoder), works for unsupervised link prediction and semi-supervised node classification. In the same year, Hisano et al. [28] worked on performance improvement using a simple discrete-time graph embedding approach for link prediction for both temporal cross-sectional network structures. Pan et al. [29] defines (ARGE and ARVGE) adversarial graph embedding framework and demonstrates the efficiency of the algorithm through experiments.

To reduce the information loss, Di et al. [30] recently presented an approach to expand the normal neighborhood when aggregating GNNs. This approach is suitable for graph link prediction, supervised and semi-supervised graph classification, and graph edge classification. Recently, Zhang et al. [31] have advanced research in link prediction using the SegNMF method. This method claims to provide better accuracy in temporal link prediction than the previously developed method.

All the state of art methods discussed above take into account the spatial embeddings of the node into account where the nodes are selected randomly for training the model. Further, the test data taken for predicting the accuracy of the model for link prediction task is very small (5 - 10%). Further few recent state of art models proposed for link prediction task in [27], [32], [33] are designed for solving problems of specific domain only. The complexity of these models tend to increase with the increase in the size of the network. So, the models do not guarantee to generalize well for networks of different nature, size and domain. Thus, the applicability of GNNs for this task on various problems in different domains can still be improved and extended. In summary, following gaps are identified and these gaps motivate us to propose the solution:

- There are no/limited approaches for predicting links between nodes in a graph with limited information available for training the network [34], [35].
- There is no centrality-based approach that can improve the prediction capability of GCN model to identify connections between nodes.
- There is a need for a generalized model which is dependent upon the structural aspects of the underlying network and independent of the application [27], [32],[33].

III. Bet-GCN Approach to Link Prediction

This section discusses how *edge betweenness centrality measure* in combination with *Graph Convolutional Network* (*GCN*) enhances the task of predicting links between unconnected nodes of the network. The content of this section has been divided into the following subsections:

- Basics of edge betweenness centrality.
- Link prediction as a binary classification problem.
- Justification of edge betweenness based training set selection.

A. Basics of Edge Betweenness Centrality

The concept of network was proposed by Roethlisberger et al. [36]. This concept defines the importance of a node based on various attributes such as the degree of a node, closeness with the nodes in its neighborhood, the number of nodes for which it is central, etc., i.e., it identifies the potential of the underlying node in terms of guiding and channeling the flow of information in the network. Based on this, there can be several *centrality measures*, e.g., degree centrality, closeness centrality, PageRank and hits centrality and **betweenness** centrality, etc. In the paper by Saxena and Jadeja [37], all these centrality measures are discussed in detail. Moreover, we investigate the suitability of the centrality measures to find out important nodes depending on the problem or task. In this section, we restrict ourselves to the betweenness centrality measure. The interconnectedness centrality measure is a centrality measure based on the shortest path. Thus, the importance of a node is recognized based on the maximum number of shortest paths in which it participates. This path-based measure, proposed by Freeman et al. [38] has two conjectures: i) *node betweenness* ii) *edge betweenness*. However, one is the implication of the other. The notion of edge betweenness centrality suggests that an edge is involved in the maximum number of shortest paths. Looking at the Fig. 3, the edge **AB** has the highest betweenness centrality compared to other edges in the network. This is because the edge AB is part of most shortest paths between any pair of vertices of the given graph. Consider two sets: set $X = \{A, F, G, H\}$ and set $Y = \{B, C, E\}$. All shortest paths from any vertex of set X to any vertex of set Y use edge AB.

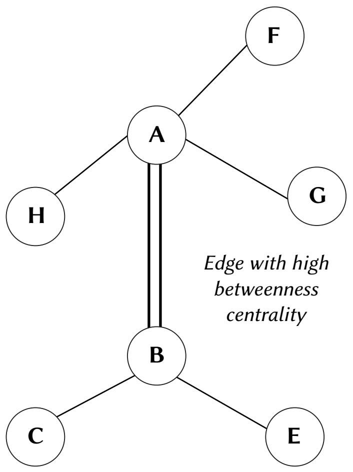


Fig. 3. Graphical network with edge AB as high betweenness centrality edge.

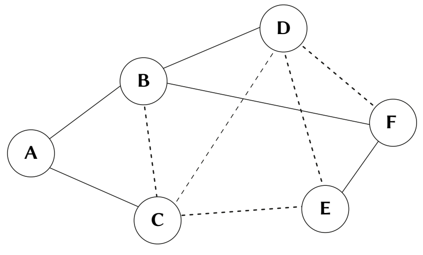
Formally, to identify the *betweenness centrality* of node *x*, we have:

$$c_{bet(x)} = \sum_{y,z\neq x,\sigma_{yz}\neq 0} \frac{\sigma_{yz}(x)}{\sigma_{yz}} \tag{11}$$

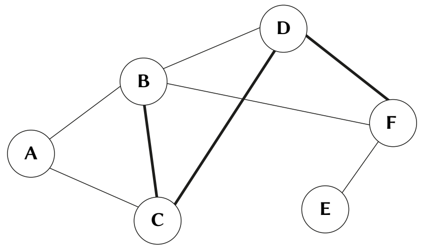
Here σ_{yz} is the total number of shortest paths leading from y to z, and $\sigma_{yz}(x)$ refers to the number of these paths that pass through x. Thus, the more shortest paths emanating from node x, the more central node x is. Edges that have one of these nodes as an endpoint have high edge *betweenness centrality*. Edges with high betweenness centrality are especially important in a large network. Endpoint nodes of an edge with high betweenness centrality are more reachable in the network with shorter path lengths. Thus, this property allows us to take advantage to increase node coverage. We use this concept to improve the performance of the GCN. An in-depth analysis and execution of this concept is presented in Section IV of the paper. In the following subsection, we discuss the approach to link prediction in a given network as a binary classification problem.

B. Link Prediction as a Binary Classification Problem

The task of link prediction is to determine whether or not a pair of nodes will have a link between them in the future. Consider a graph $G(V, E)$ at a given time t with V as a set of nodes and E as a set of edges, as shown in Fig. 4a. The graph shows various possible links between pairs of nodes that can occur at time $t + \delta t$ (represented by dotted lines). At time $t + \delta t$, as shown in Fig. 4b, some expected connections appear (shown by bold edges in the graph), while some of them do not. Graph Convolution Networks (GCN) captures the properties of the nodes in addition to the topological and structural information of the network. This helps in finding close correlations and probable neighbors of a node based on their behavioral similarities in the network. However, to do this, we must first model the problem in a structure of < *feature, target* > pairs to apply a graph-based machine learning model.



(a) Graphical network at time t



(b) Graphical network at time $t + \delta t$

Fig. 4. Evolution of Graph G from time t to time $t + \delta t$.

Each node has a feature set (vector) associated with it. It consists of a collection of information about the node, its properties, etc., that define and identify that node in the network. An edge has two nodes as its endpoint, so the final feature set in this case is a combination of the feature vectors of the two nodes that form the edge. If we consider the edge as (u, v), where u and v are the nodes under consideration, the final feature vector is as follows:

$$\textit{feature vector} = \textit{feature vector}(u) \cup \textit{feature vector}(v)$$

Further, depending upon whether the two nodes u and v are connected or not, target (label) can be defined as:

$$\text{target} = 1 \text{ , if } (u, v) \text{ is connected}$$
$$= 0 \text{ , otherwise}$$

Thus, by separating the connected and disconnected nodes with the labels 1 and 0, respectively, we can create a *pair* (see Table I). It is now possible to process the data with a machine learning model to make predictions. For the given graph G, we can select a pool of edges for training the GCN model. Here, each edge is accompanied by its label. Also, the GCN model uses the node feature information to train the model. The edges of the test dataset can be randomly selected to test the accuracy of the model for the binary classification problem, i.e., predict 1 for each connected pair and 0 for each unconnected pair.

TABLE I. Graph Edges With Labels

Connected Edge	Label	Unconnected Edge	Label
A-B	1	B-C	0
A-C	1	C-E	0
B-D	1	C-D	0
B-F	1	D-F	0
E-F	1	E-D	0

For real networks, the model is created by randomly hiding some edges from the network. The remaining network is then used to train the GCN. The hidden edges are then used to test the adequacy of the model. This simulative technique is as good as analyzing the temporal transition of the graph because: i) we do not have timestamp snapshots of the real networks at persistent intervals and ii) the network changes its structure gradually. Thus, the network is not significantly perturbed. For these two reasons, we consider only a single real graph as input. In the following subsection, we discuss and analyze how *edge betweenness centrality* based training set selection improves the efficiency of the GCN model for link prediction.

C. Justification of Edge Betweenness Centrality Based Training Set Selection

So far, we have discussed the *edge betweenness centrality* measure and the strategy for solving the link prediction. In this subsection, we will analyze the basis of our proposal:

Training set selection based on edge betweenness centrality improves GCN training efficiency. For this purpose, let us consider a graph $G(V, E)$ for which holds:

V: Set of vertices or nodes defined as $\{v_1, v_2, ..., v_n\} \in V$

E: Set of edges or links defined as $\{e_1, e_2, ..., e_k\} \in E$ such that $n, k > 1$

Let X be the feature matrix defined as:

$$X = \{\{x_{11}, x_{12}, \dots, x_{1m}\}, \dots, \{x_{n1}, x_{n2} \dots x_{nm}\}\} \tag{12}$$

In general, we have $n > m$ (size of training data (number of nodes) > length of a feature vector) to avoid the condition of overfitting during the training process.

Now a set of edges is chosen from the set E to generate a test set t_1 containing t_1. The t_1 is a subset of E containing all connected pairs of nodes. For all these edges (or node pairs), the class label set l_1 is defined as 1. Now, a few random unconnected node pairs are selected from the set $Complement(E)$ or $\overline{E}$ to generate another test set t_2. The corresponding label set for the node pairs of the set t_2 is defined as l_2 with label value 0. Combining the test sets t_1 and t_2, the final test set t can be defined as:

$$t = t_1 \cup t_2 \tag{13}$$

Corresponding to it, the label set L for this test set t can be defined as:

$$L = l_1 \cup l_2 \tag{14}$$

Deleting edges from the graph G creates a graph G', where the edge set of G' is defined as $E' = E - t$. From this residual graph, the training set is constructed in the same way as the test set. Based on this training set, the predicted set of labels for the edges selected from the test set t is obtained as L'. Thus, the objective of the problem can be formulated as follows:

(1) To obtain predicted label set L', we use the GCN model for the edges in test set t, which approximates the label set L i. e., Min. $(L' - L)$ ∀ edges in t.

(2) With respect to identification of such a subset, the following observations are made:
- The subset of edges (or node pairs) selected based on the betweenness centrality measure improves the training efficiency of the model.
- The probability of random selection of such an edge set to produce a predicted label set L' is nearly zero.

(3) Let us try to infer the validity of the first statement.
- E'' contains subset of edges chosen randomly. Let this subset be named as E_1.
- E'' contains top d edges based on the betweenness centrality score. Let this subset be named as E_2.

Further, it is assumed that $Cardinality(E_1)$ and $Cardinality(E_2)$ are the same. Let us consider the first edge from each subset. Let a be the edge chosen from E_1 and b be the edge chosen E_2. Let σ_b represent the edge betweenness centrality of node b and σ_a refer to the edge betweenness centrality of node a. Thus, it is obvious that:

$$\sigma_b > \sigma_a \tag{15}$$

Further, we also assumed that the edge sets E_1 and E_2 are disjoint, i.e., no edges are common to the two sets. Then, extending the above expression for $1 \le i \le l$:

$$\sum_{i=1}^{l} \sigma_{(b_i)} > \sum_{i=1}^{l} \sigma_{(a_i)} \tag{16}$$

Equation (16) holds for a fixed path length p, for the paths covered by the edges in E_1 and E_2. As can be seen from the description of GCN in Section II, it is well known that the neighborhood contribution beyond path length 2 or 3 is not beneficial because of the vanishing gradient problem over the graph Laplacian. So the value of p is $\in \{1, 2\}$. As per Section III, edges with high betweenness centrality allow for greater network coverage with shorter path length. This means that the node coverage (number of reachable nodes) from the nodes of the set E_2 (say ϕ) will be larger than the number of reachable nodes from the nodes of the set E_1 (say α), i.e.,:

$$\phi > \alpha \tag{17}$$

For a GCN model, training efficiency (η) depends on feature availability (f.a.), i.e., the more features available to the model for learning, the better the training of the model. Feature availability increases when the number of nodes reachable from a fixed set of nodes is high, since each node is associated with a feature vector X. Thus, feature availability again depends on node coverage or node reachability (κ). Based on all these discussions, a relationship can be established that looks like the following:

$$\eta \propto f.a. \propto \kappa \tag{18}$$

Considering equations (17) and (18) synchronously, the set E_2 will cover more neighborhood nodes, which means greater feature availability since each node is associated with a feature vector X. This increases the training efficiency of the model compared to selecting the training set based on E_2. To test this observation empirically, let us consider a small example according to Fig. 5. Consider $E_1 = \{(A, F), (B, C)\}$ as the edge set selected for training. For a fixed path length 2, the node coverage of the set is E_1:

$$\alpha = \{A, B, C, D, E, F, G, H\} \tag{19}$$

Now consider another edge set $E_2 = \{(A, B), (E, D)\}$ where the two edges with high betweenness centrality value are selected for training. For the same path length 2, t,he node coverage is the same for this training set:

$$\phi = \{A, B, C, D, E, F, G, H, I, J, K, L, M\} \tag{20}$$

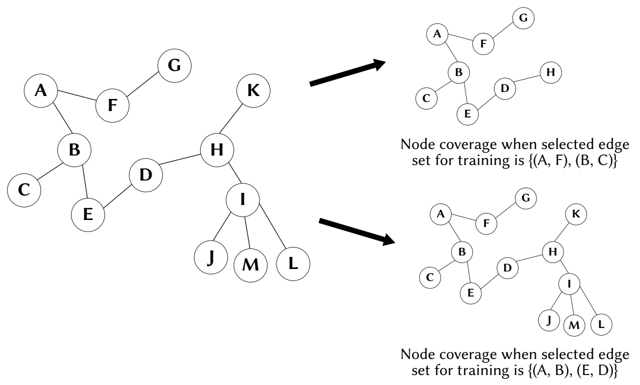

A
B
C
D
E
F
G
H
I
J
K
L
M
Node coverage when selected edge set for training is {(A, F), (B, C)}
Node coverage when selected edge set for training is {(A, B), (E, D)}

Fig. 5. Node coverage of graph $G(V,E)$ based on training edge selection.

Since E_2 has a larger number of nodes in its neighborhood, the availability of features will also be larger. And finally, it can be confirmed that the training efficiency of the model improves. Thus, it has been successfully analyzed that the selection of the training set based on the *edge betweenness centrality* improves the learning of the GCN-based training model for link prediction. On this basis, we can say that a mapping L' can be obtained which is approximately equal to L.

In the proposed method, the training is edge based, not node based. Hence, the criteria of edge set selection based on the betweenness centrality of edges makes sense. On the other hand, edge selection based on nodes having high degrees is not feasible. The reason for this is a high degree node has many edges associated with it. Each edge associated with the node will have equal weightage. Hence, all the edges incident on the high degree vertex will be selected for training. In such a situation, the model may miss out a significant portion of the network required for training since only edges which are incident to the high degree vertices will be selected. Clearly, this selection fails to capture the crucial structural properties of the of the network. Also, this degree-based selection will not allow the training set to capture diverse feature vectors which is essential for efficient training of the model.

On the other hand, consider the betweenness centrality-based approach for edge selection as discussed in *subsection* B of section III. This high betweenness edge centrality based selection will lead to generation of computation graphs with more number of nodes (in average) during training. Since, feature set aggregation is directly proportional to number of nodes in the underlying computational graph, a better training of the GCN model is guaranteed using proposed approach. This in turn enhances the prediction capability of the model.

Next, we need to ensure that the probability of randomly selecting the edge set E_2 is close to zero. Let us consider the total number of edges in the network as k, such that $k > 1$. The number of ways to choose a subset of length w (subset of w edges) is given as ${}^kC_w$. Our goal is to find the probability of choosing the subset E_2 from these ${}^kC_w$ subsets. Thus, let us consider an event Q as: *choosing the subset E_2 of the set* E, where E is the set of all edges of the graph such that $|E| = k$. The probability of this event will be:

$$P(Q) = 1/{}^kC_w \tag{21}$$

Let us assume that 45% of the edges are used for training. Thus, we have $w = (9/20)k$. Putting this value of w into the equation (21), we get,

$$P(Q) = 1/{}^kC_{9k/20} \tag{22}$$

In general, the number of edges for real network graphs is on the order of more than 10^4. Plugging the value of k as 10^4 into the expression, we get,

$$P(Q) \approx 0 \tag{23}$$

Finally, we can also successfully show that the probability of randomly choosing the edge set E_2 is close to zero. Thus, the section successfully verifies the two arguments: i) selecting the training set based on edge centrality improves the performance of the model. ii) the probability of randomly selecting an ordered set based on the centrality score is close to zero. In the next section, we detail the proposed method and its design along with the description of the dataset.

IV. Dataset and Model Description

In this section we discuss mainly about the datasets, the proposed model formulation, and aspects related to its implementation.

A. Dataset Description

To assess the performance of the proposed model, three famous state of the art datasets have been chosen: *CORA*, Citeseer and *PubMed*. The datasets have been summarized in Table II.

TABLE II. Dataset Description

Dataset	Nodes	Edges	Classes	Features	Type
Cora [25]	2,708	5,429	7	1,433	Citation Network
Citeseer [25]	3,312	4,732	6	3,703	Citation Network
PubMed [25]	19,717	44,338	3	500	Citation Network
Amazon [39]	13,752	491,722	10	767	Amazon Product Network
WikiCS [40]	11,701	216,123	10	300	Wikipedia Network

The first three datasets considered are essentially citation networks where *node* stands for *papers* and *edge* stands for the *citation links*. The CORA citation network consists of 2708 scientific publications classified into one of the following seven classes: neural networks, rule learning, reinforcement learning, probabilistic methods, theory, genetic algorithms, and case-based. For each node, there is a feature word vector of length 1433. Thus, the size of the feature matrix is 2708 × 1433. The Citeseer dataset consists of 3312 scientific papers classified into six classes: Agents, AI, DB, IR, ML, and HCI. The feature matrix has order 3312 × 3703. The PubMed citation network consists of 19,717 scientific publications with the following classification classes: *1, 2, 3* i.e., diabetes type-1, 2 and 3. The feature vector for each node consists of a TF/IDF vector with 500 unique words. The accuracy of the proposed model with GCN-based training was tested using these three benchmark datasets. The consistency of the results obtained with these networks highlights the effectiveness of the proposed solution. To prove the applicability of the proposed solution to other types of networks, two other graphical networks are considered. *Amazon Computer* [39] is a segment of the Amazon co-purchase graph, which is a network collected by crawling the Amazon website and contains product metadata and rating information about various products. The nodes in the graph represent items, while the edges indicate that two or more goods are usually purchased together. The goal is to assign items to the appropriate product categories by using product ratings as node attributes. *WikiCS* [40] is a novel dataset derived from Wikipedia to benchmark Graph Neural Networks. The dataset contains 11701 nodes corresponding to computer science articles, with edges based on hyperlinks, and 10 classes representing different branches of the field.

It is common for real-life applications with graphs to have limited training data because labels will often be sparse, despite having vast quantities of data. This is true for all the datasets considered in this manuscript. Hence, they are limited training datasets. In the context of link prediction, labels are edge labels (0 for not edge and 1 for an edge). And for training, a very small fraction of labels are known. For example, for Cora, labels of only 5429 edges are known (label 1) out of 3665278 possible edges. Labels of the remaining 3659849 edges are unknown. Hence, the Cora dataset is a limited training dataset. The same is true for other datasets too as shown in Table III.

TABLE III. Dataset With Actual V/s Possible Eddges in the Graphical Networks

Dataset	Nodes	Total possible edges	Total edges in actual graph
Cora	2708	3665278	5429
Citeseer	3312	5483016	4732
PubMed	19717	194370186	44338
Amazon	13752	94551876	491722
WikiCS	11701	68450850	216123

Following this, the next subsection explains the implementation design and operation of the proposed model.

B. Proposed Framework and Experimental Setup

To construct a Graph Convolution Network based training model architecture, *Stellar Graph library* [41] was used. In addition, the graph library *NetworkX* [42] is used to capture the structural information of the network. The input data set for the GCN model consists of an edge list and a *feature matrix* along with *labels.*

The relationship that exists between the data points (nodes) of the graph is represented by the *links* between them, defined by the *edge list*. To prepare the **test dataset**, an *Edge Splitter* () function from the *Stellar Graph library* was used. This function randomly takes some pairs of nodes from the original graph *G*. For each connected pair, the associated label is 1. Also, some unrelated pairs are randomly selected and these pairs are assigned the label 0. Thus, we obtain the final test set tuple *t* for which the label set *L* is defined with labels 0 and 1 for each unconnected and connected pair of nodes, respectively.

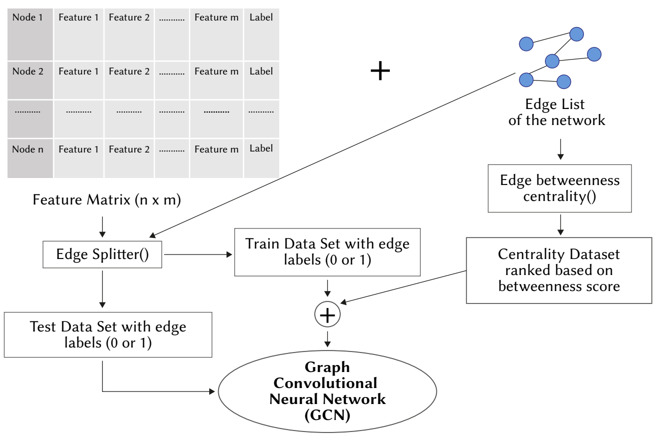
Let us now consider the training dataset. After removing the edges in the test set t from the graph *G*, the training dataset is selected from the residual graph *G'*. The training dataset contains the top *k* edges with high values of betweenness centrality computed using the *NetworkX* graph library (nx.edge_betweenness_centrality()). This part of the training dataset is denoted as *tr*1 with the corresponding label set as *trl*1 with all label values as 1. Furthermore, few edges are sampled using the *edgesplitter*() function to include some unconnected pairs. Let this part of the training dataset as *tr*2 with the label set *trl*2. Thus we have the final training dataset defined as:

$$tr = tr1 \cup tr2 \tag{24}$$

with training label set defined as:

$$trl = trl1 \cup trl2 \tag{25}$$

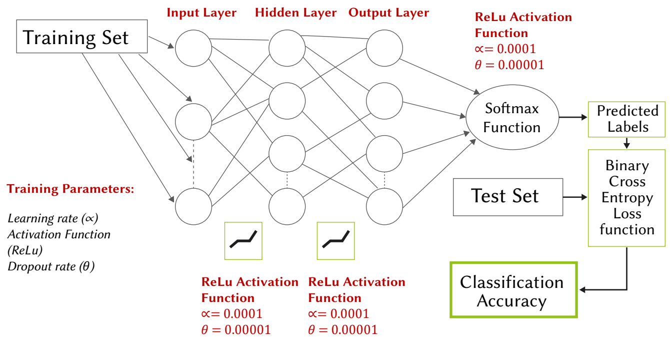
Fig. 6 explains the steps to generate a training dataset (55%) and a test dataset (upto 45%). The input graph dataset consists of edge list information along with node feature vectors. Note that each node has a feature vector associated with it. To create the test dataset, edge splitter function randomly pools the edges, marked as label '1', and an equal number of node pairs amongst which no direct edge exists, marked as label '0'. A similar procedure is adopted by the function to create the train dataset. However, in addition to the edges selected, top 'k' betweenness centrality metric-based edges are also appended in the training dataset. Finally, the train and test datasets are supplied to the GCN model. Since the connectivity of the graph must be ensured, it is not possible to extract a very high percentage of edges from the graph for the creation of the test dataset. Therefore, the test dataset here is a combination of validation test dataset.



Node 1 | Feature 1 | Feature 2 | .... | Feature m | Label
Node 2 | Feature 1 | Feature 2 | .... | Feature m | Label
Node n | Feature 1 | Feature 2 | .... | Feature m | Label
Feature Matrix (n x m)
+
Edge List of the network
Edge betweenness centrality()
Centrality Dataset ranked based on betweenness score
Edge Splitter()
Train Data Set with edge labels (0 or 1)
Test Data Set with edge labels (0 or 1)
Graph Convolutional Neural Network (GCN)

Fig. 6. Proposed Framework: Feeding train and test set to GCN.

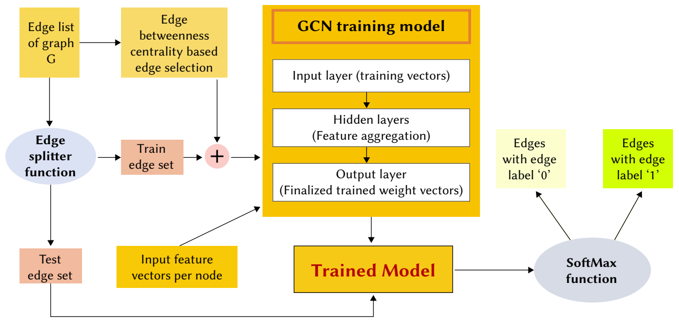
Using the node feature matrix defined for each node for the nodes involved according to the training set selection, the model is fed with the input. The function *FullBatchnode Generator*() defines the neural network (NN) for the graphical network. The defined neural network has three layers: the input layer, the hidden layer, and the output layer. The number of hidden layers is best determined from the experimental simulations. However, in the case of GCNs, the number of hidden layers corresponds to the diameter of the graph. This refers to the number of neighbors that are a path length k away from the node under consideration. The value of k is generally kept very low because the vanishing gradient problem affects the performance of the model. Other hyperparameters of the model such as *kernelsize*, *learningrate*, *epochs*, *activationfunction*, etc. are chosen to minimize the error. The hidden layers have a Rectified Linear Unit (ReLU) activation function with a hidden layer size on the order of $4,096 \times 4,096$. However, the size of the *kernel* varies depending on the size of the network. Other parameters of the network such as *learning rate* is set to 0.0001 with *Adam Optimizer* and *Cross Entropy* as loss functions. The output layer of the model uses a *Softmax* function to predict the presence of an edge between a pair of edges over the test dataset. Fig. 7 explains the GCN-based training and classification process.



Training Set
Input Layer
Hidden Layer
Output Layer
ReLu Activation Function ∝= 0.0001 θ = 0.00001
Softmax Function
Predicted Labels
Binary Cross Entropy Loss function
Test Set
Classification Accuracy
Training Parameters:
Learning rate (∝)
Activation Function (ReLu)
Dropout rate (θ)
ReLu Activation Function ∝= 0.0001 θ = 0.00001
ReLu Activation Function ∝= 0.0001 θ = 0.00001

Fig. 7. GCN Based Training of the model.

Fig. 8 sums up the entire process in a block diagram. The algorithmic steps in training of Bet-GCN model are as shown in algorithm *Bet-GCN*. The input to the model is an input graph dataset $G(V, E)$ where V represents set of vertices and E represent set of edges. Each node in the graph has feature vector associated with it. Let A be the adjacency matrix for the graph and X be the feature matrix (as mentioned in equation 12). The algorithm will yield a trained model m which can predict whether an edge exists (edge label 0) or not (edge label 1) between two given pair of nodes (binary classification problem). In **step 1**, *edge splitter* function randomly pools a set of edges from graph G to prepare training dataset (say *Tr*). The training set consist of edges which exist in the graph labelled as 1 and edges which do not exist in the graph labelled as 0. In **step 2**, from the remaining graph (say G'), *edge splitter* function constructs the test dataset (say *Te*) in a similar manner. **Step 3** and **Step 4** identifies the top k edges in order of *edge betweenness centrality*. The top k edges identified in *step 4* are added in **step 5** to *Tr* to generate the final training dataset. In **step 6**, the GCN model is fed with *Tr*, G and the model hyperparameters like *learning rate, layer size, ReLu activation function*. The input layer is fed with an aggregation function defined as *A.X*. The hidden layer further performs feature aggregation using a layer size $4,096 \times 4,096$ at a learning rate 0.0001. The ReLu activation function is applied to obtain the convoluted vector (neighborhood aggregation) matrix at each layer. At each layer gradients are determined and based upon the error function gradients, using backpropagation algorithm weights are adjusted. This whole process iterates till the error gradient functions at each layer evaluates out to be zero. In this condition, we obtained a finalized weight vector matrix at output layer and the trained model m. Finally, in **step 7**, the trained model is tested over *Te* using *SoftMax()* classification function to generate the classification report.



Edge list of graph G
Edge betweenness centrality based edge selection
GCN training model
Input layer (training vectors)
Hidden layers (Feature aggregation)
Output layer (Finalized trained weight vectors)
Edge splitter function
Train edge set
Edges with edge label '0'
Edges with edge label '1'
Test edge set
Input feature vectors per node
Trained Model
SoftMax function

Fig. 8. Pictorial block diagram for Bet-GCN model.

V. Results and Analysis

This section mainly focuses on the experimental results and performance of the proposed Bet-GCN model. It highlights the significant results of the model in three benchmark datasets, namely Cora, Citeseer and PubMed, and the comparative analysis with the respective state-of-the-art methods. The results of our proposed model Bet-GCN (Edge betweenness centrality with Graph convolutional networks) are summarized in Table IV. The performance of the model improves considering that the model performs well on a large test data set. All of the state-of-the-art methods discussed work over 5 10% test data. The Bet-GCN based results are analyzed over upto 45% test data with at least 30% unseen node pairs in the test dataset.

TABLE IV. Citation Networks Accuracy

Method	Cora	Citeseer	PubMed	Test Dataset
VGAE [25]	0.920	0.914	0.965	-
MTGAE [27]	0.946	0.949	0.944	5-10%
GLP [32]	0.9455	0.8612	-	5-55%
GCN [33]	0.9050	0.8701	0.9694	-
GAT [33]	0.8979	0.8731	0.9436	-
EdgeConv [33]	0.8528	0.8294	0.8665	-
EdgeConvNorm [33]	0.9178	0.8754	0.8991	-
Bet-GCN(proposed)	0.9508	0.9507	0.953	upto 45%

```
Algorithm 1. Bet-GCN
Require: An input graph dataset $G(V, E)$ where $V$ represents set of vertices and $E$ represent set of edges
Output: The trained model, $m$
Step 1. $Tr \leftarrow edgesplitter(G)$
    ▷ Generating training set $Tr$ by random selection of edges from graph $G$
Step 2. $Te \leftarrow edgesplitter(G)$
    ▷ Generating test set $Te$ by random selection of edges from graph $G$
Step 3. $e \leftarrow edge\_betweenness\_centrality(G)$
    ▷ Evaluating edge betweenness centrality of edges of graph $G$
Step 4. $e' \leftarrow sorted(e[1{:}k])$
    ▷ Selecting top $k$ edges based on edge betweenness centrality of edges of graph $G$
Step 5. $Tr \leftarrow Tr \cup e'$
    ▷ Adding edges form step 4 to $Tr$
Step 6. $m \leftarrow GCN(G, Tr, learningrate, layersize, ReLu)$
    ▷ Obtaining the trained GCN model $m$
Step 6. $classication\_report \leftarrow SoftMax(m, Te)$
    ▷ Testing the model over test set and generating classification report
```

As summarized in Table IV, most models consider methods such as random walks (where only local node similarity is used) or maximum likelihood estimation methods for link prediction. It can be observed that none of these methods materialise the node features, the structure of the underlying network, or the importance of the edges completely. In comparison, GCN, which considers the structure of the dataset as a graph, significantly improves link prediction performance. Traditional Graph Convolutional Networks (GCN) directly convolve the structure of the connected graph as a filter to perform neighbourhood mixing. Graph Attention Networks (GAT), on the other hand, apply a shared linear transformation to each node, followed by a computation of attention coefficients using a joint attention mechanism. The performance of link prediction with these two models is impressive and promising. A more recent state of the art, the Variational Graph Auto-Encoder (VGAE), uses a graph convolutional network as an encoder that maps the node features into a latent representation, followed by a decoder that generates conditional probabilities of the adjacency matrix [25]. While Multi-Task Graph Autoencoders (MTGAE [27]) learns a joint representation of latent embeddings from a local graph and explicit node features. These two methods are significantly better than the traditional GCN model due to the inclusion of autoencoders. In addition, GLP [32], a gravitational link based unsupervised approach is used. Here, the main idea is to decompose the graph into a local structure (by extracting subgroups) and a global structure (by detecting communities). The method showed promising results on large complex networks, but is highly dependent on the network structure. Two recent link prediction methods based on Graph Convolution Learning were also proposed: *EdgeConv* and *EdgeConvNorm* [33]. The methods performed well on the three networks, as the over-smoothing of Edge Convnorm helps to better learn link prediction based on the node and its neighborhood representation.

Our proposed model (Bet-GCN) is a modification of the traditional GCN model, as it uses edges based on their betweenness centrality in the graph along with node features. Our proposed prediction model achieves an accuracy of 95.08% in Cora, 95.07% in Citeseer, and 95.32%in PubMed. These results are competitive with the current state-of-the-art models, which can be observed in Table IV.

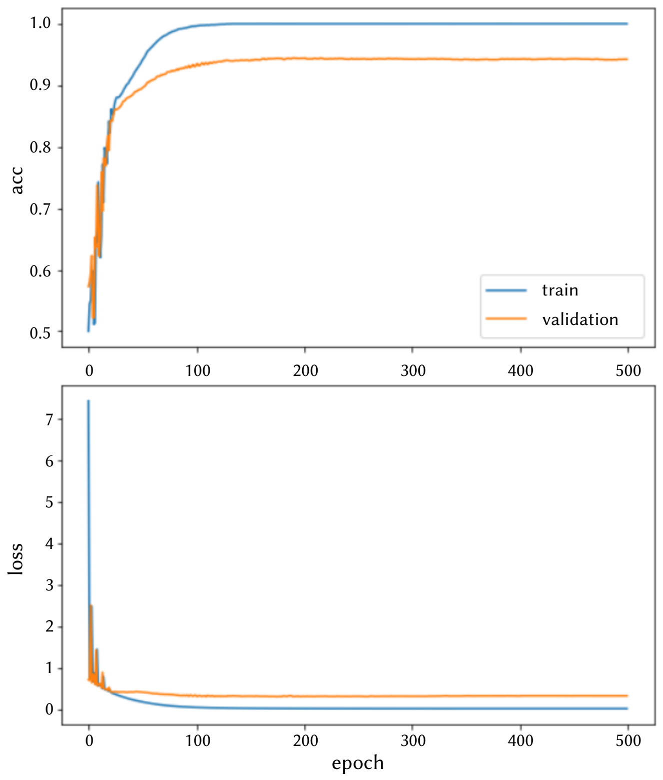


Fig. 9. CORA: Training and Loss curve.

The model extrapolates the structure of the underlying graph for sampling positive edges when training the model for prediction. BET-GCN architectural hyperparameters were fine-tuned for the Cora, Citeseer, and PubMed networks. A 0.70 and 0.35 fraction of the original network is randomly sampled for positive and negative edges as training and test edges, respectively. The positively sampled training edges are replaced with the edges sorted based on the edge betweenness centrality score. A two-layer GCN model is used, where 4, 096 is the dimension of the node features in each hidden layer. The Rectified Linear Unit (ReLU) activation function is used. For the final link classification, a pair of node embeddings from the GCN model is used and the binary operator inner product (ip) is applied. This produces the corresponding link embedding, which is passed through a dense layer. A learning rate of 0.0001 for Adam Optimizer is used to train the model. Our model is trained with 500 epochs. These hyperparameter settings are the same for Cora and Citeseer citation networks. The PubMed dataset consists of 10x more edges and therefore has different hyperparameter values. The training accuracy and loss curves of the model for the three datasets are shown in Fig. 9, Fig. 10, and Fig. 11, respectively. Based on the obtained results, it can be confirmed that the proposed method performs best for the three collaboration networks.

Area Under the Curve (AUC) curves of the model obtained for the three datasets are shown in Fig. 12, Fig. 13 and Fig. 16. The AUC curves show the ability of the classifier to distinguish correctly between positive and negative classes. The high AUC value for all three datasets CORA (94.02%), Citeseer (94.24%), and PubMed (97.96%) indicates the consistency of the model in terms of performance.

The hyperparameters' settings depend upon the size and structure of the network for training GCN models. The basic parameter settings' have been considered based upon Thomas N Kipf and Max Welling [1] paper. The parameters that are varied are layer size, learning rate and iterations due to varied network structures and sizes. In general, ReLu activation function has been used for $4,096 \times 4,096$ layer size

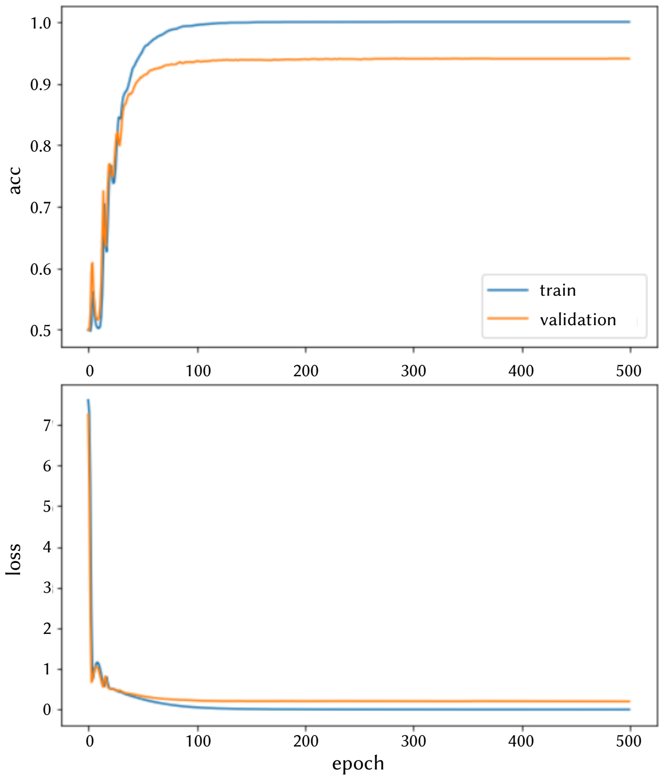


Fig. 10. Citeseer: Training and Loss curve.

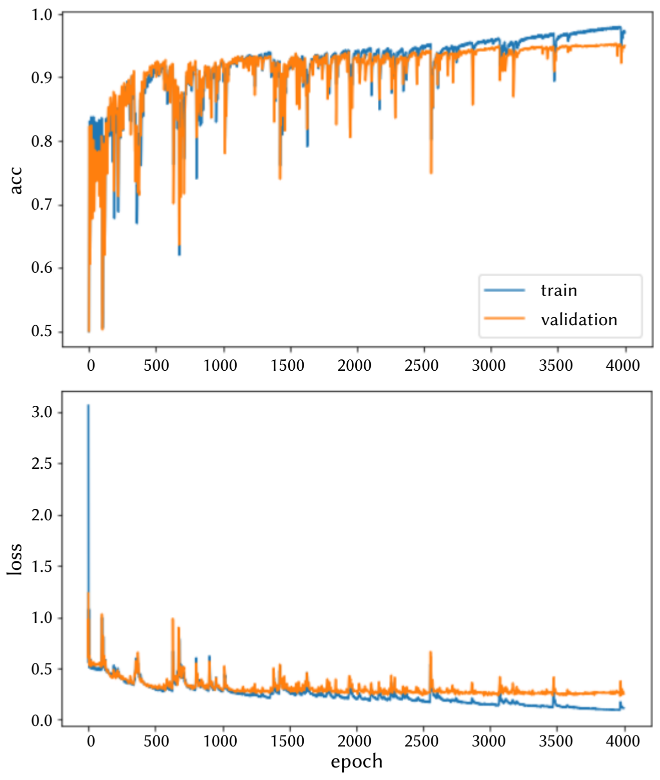


Fig. 11. PubMed: Training and Loss curve.

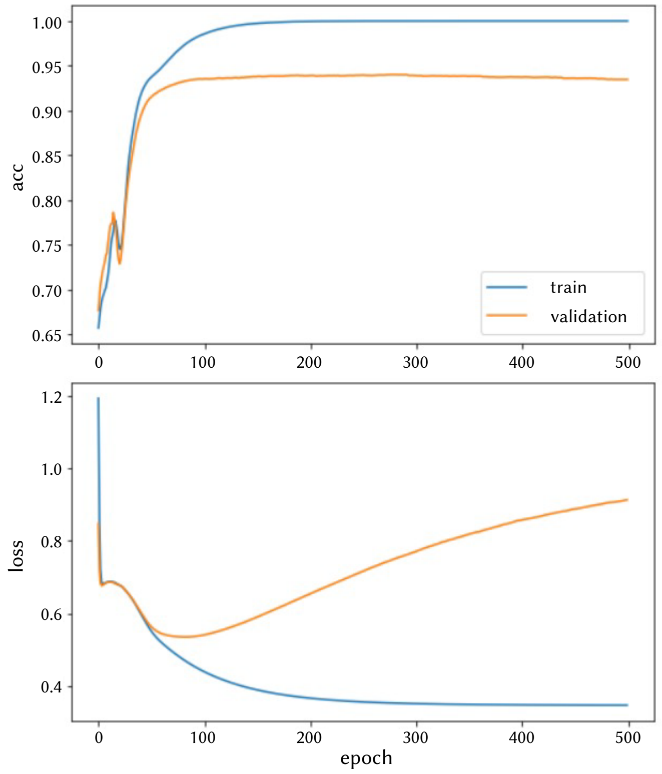


Fig. 12. AUC Curve for CORA network with accuracy (94.02%).

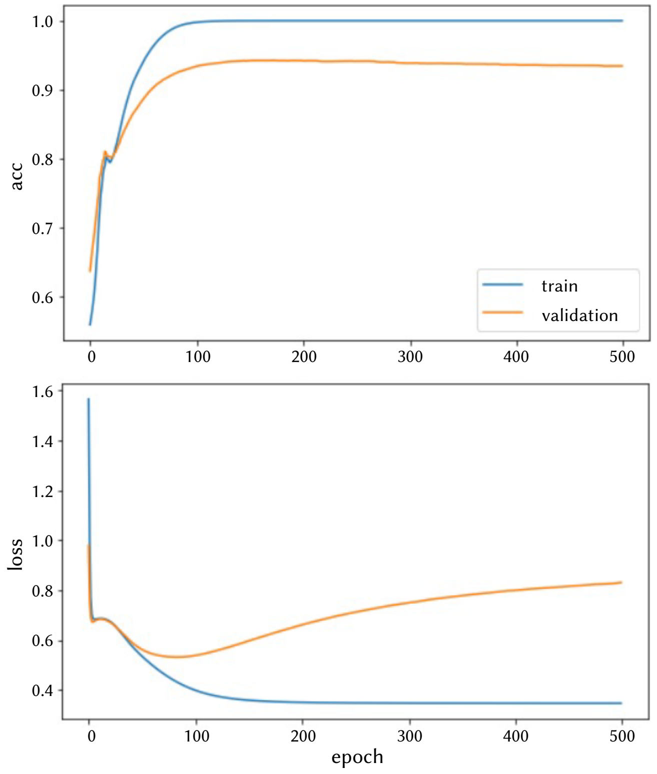


Fig. 13. AUC Curve for Citeseer network with accuracy (94.24%).

of hidden layer which yield the best results. The number of iterations is identified based upon the training accuracy curve trajectory and, hence, the number of iterations is different for each network. The iterations' convergence happens when the training accuracy starts to dip for several continuous iterations. So, further increasing the number of iterations will not yield good results and may tend the model to overfit. Similarly, the best results are obtained for a learning rate of 0.0001 for the three benchmark datasets into consideration (CORA, Citeseer and PubMed). Fig. 14 shows that further reducing the learning rate causes a drop in the performance efficiency of the model for CORA dataset. Fig. 15 presents the trend analysis of the performance of the model with number of epochs. At 500 epochs, keeping the learning rate fixed at 0.0001 and hidden layer size of 4096 4096, the performance attained by the model is optimum. Further increasing the number of epochs for model training is not helping the cause and the performance tends to deteriorate as the model starts *overfitting*. A similar analogy can be drawn for the size of hidden layer. Further, a similar kind of analysis can also be obtained for the two other kind of networks (Citeseer and PubMed). Thus, it can be inferred that learning rate of 0.0001 and hidden layer size of $4,096 \times 4,096$ is suitable for networks of different variety and structural formation in order to have an efficient training through Bet-GCN model. Number of epochs to attain the optimum accuracy may differ depending on the size of the network. However, all of these parameter settings are network dependent and vary slightly depending on the nature of the task.

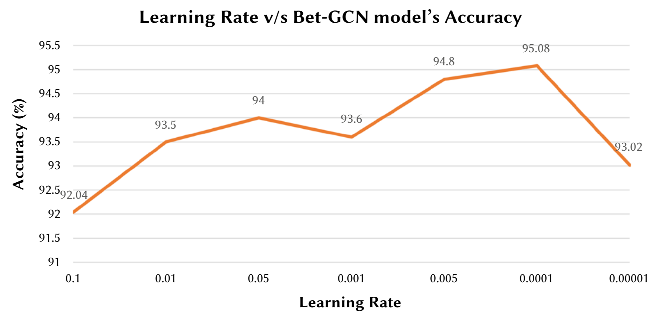


Fig. 14. CORA: Learning Rate v/s Bet-GCN accuracy curve.

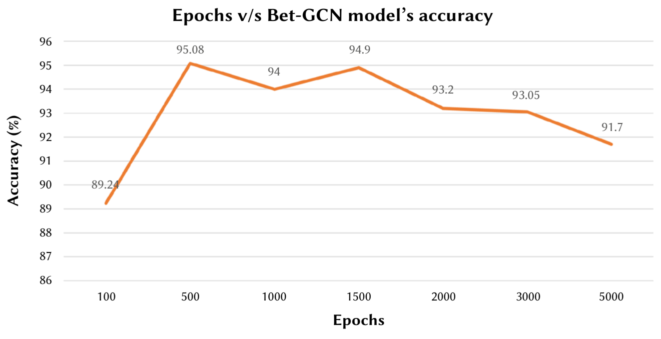


Fig. 15. CORA: Epochs v/s Bet-GCN accuracy curve.

VGAE and GAE [25] uses a Gaussian prior distribution over the input features to learn embeddings. However, this has not proven to be a very good choice. MTGAE [27] gives impressive results for link prediction, but the accuracy of the method decreases when a larger number of edges are removed from the graph. This is because only the contribution of the available edges is considered. GLP [32] involves a lot of preprocessing, such as community identification, followed by extraction of optimized subgraphs. The link prediction task is then performed over these distributed subgraphs. The method is not suitable for networks with large diameters. The other link prediction strategies mentioned in the work of Gu et al. [33] are GCN-based methods where the selection of the training set is random. Our proposed method Bet-

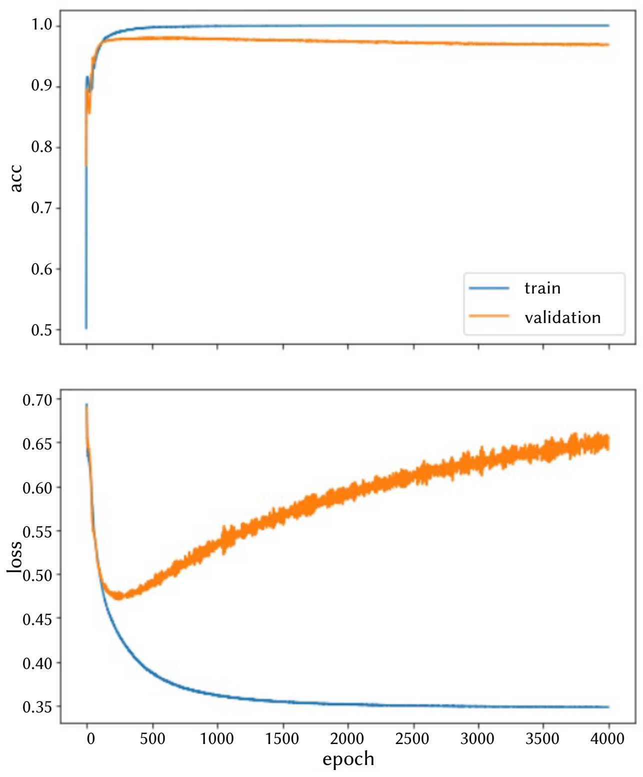


Fig. 16. AUC Curve for PubMed network with accuracy (97.96%).

GCN is also a variation of GCN technique where the training set is selected based upon the betweenness centrality score. This helps in capturing more neighborhood contribution for the model's training. As a result, there is more neighborhood aggregation in the computational graphs. This will help the model to leverage the feature-based learning and generate more accurate embeddings. Traditional GCN approaches use random selection and, hence, they are not able to capture features which are betweenness centrality based. It is due to this reason that the method performs well in comparison to the other state of art methods.

In addition, the Bet-GCN model was also tested on two different types of networks (since all three networks mentioned above were citation networks): Amazon Product [39] and WikiCS [40]. The Amazon Product network was collected by crawling the Amazon website and contains product metadata and review information for 548552 different products (Books, music CDs, DVDs, and VHS video tapes). WikiCS [40] is a web graph of Wikipedia hyperlinks collected in September 2011. Bet-GCN link prediction model for both datasets perform equally well as for the citation networks. Table V lists the accuracy and respective F1-score values for the network.

TABLE V. Accuracy and F1-score Values for Amazon Product and WikiCS Networks

Network	Accuracy	F1-Score	Test Dataset
Amazon Product	0.879	0.8801	upto 45%
WikiCS	0.9113	0.90	upto 45%

Fig. 17 and Fig. 18 show the training accuracy curves for both networks using the Bet-GCN model. The results for the network indicate that the approach is scalable with network size and applicable to graphical networks of different domains. The results for these two networks were evaluated with the same parameter settings used for CORA, Citeseer, and PubMed, except for the layer size. Fig. 19 refers to the confusion matrix for all the five graphical networks, which shows the prediction capability of the proposed model Bet-GCN. Given the

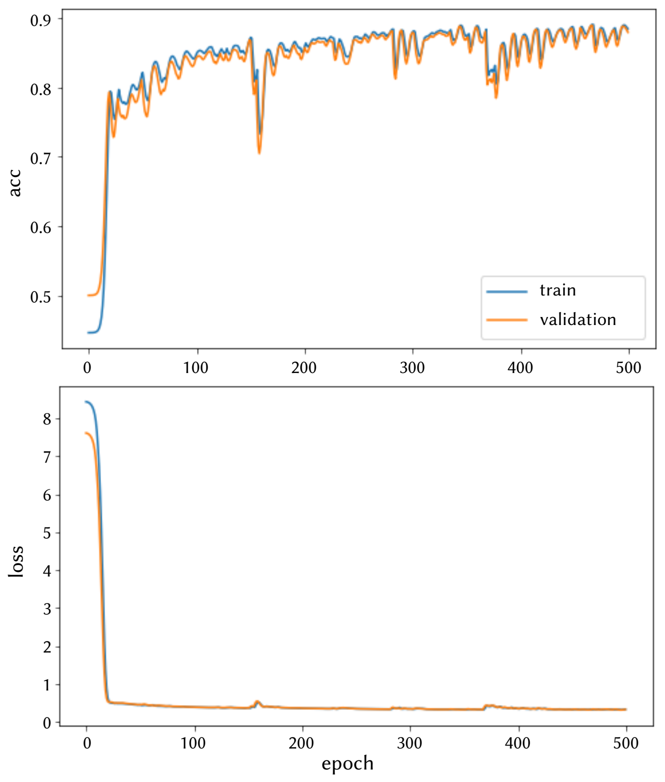


Fig. 17. Training and Loss Accuracy Curves for Amazon Product Network.

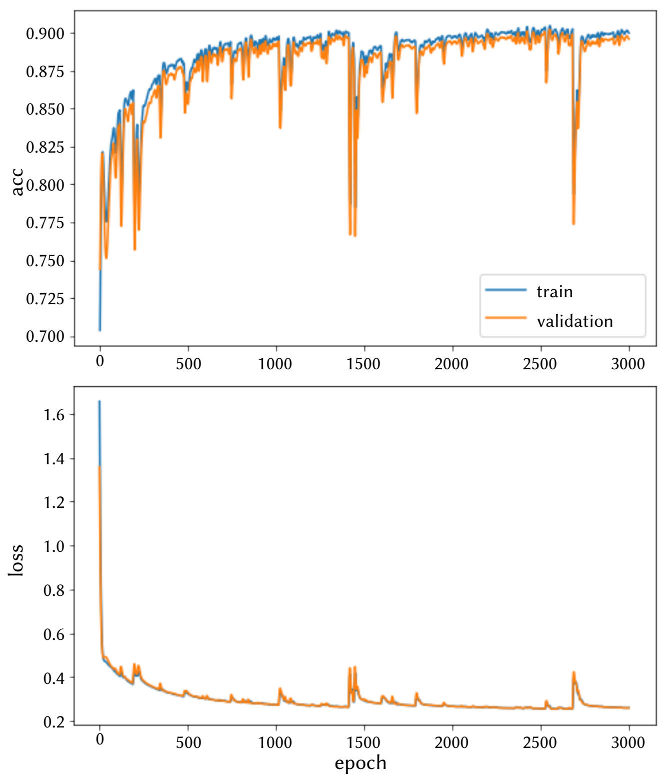


Fig. 18. Training and Loss Accuracy Curves for WikiCS Weblink Network.

CORA		
	True Positive	**True Negative**
Predicted Positive	2609	105
Predited Negative	163	2551

Citeseer		
	True Positive	**True Negative**
Predicted Positive	2233	124
Predited Negative	109	2248

PubMed		
	True Positive	**True Negative**
Predicted Positive	20356	1813
Predited Negative	344	23625

Amazon		
	True Positive	**True Negative**
Predicted Positive	277349	17684
Predited Negative	47810	247223

WikiCS		
	True Positive	**True Negative**
Predicted Positive	252262	6773
Predited Negative	39511	219524

Fig. 19. Confusion Matrix for the graphical networks.

large number of edges in the networks, the convolutional layer size used for the hidden layer is 512 × 512. As one increase the number of layers, the number of parameters that can be trained also increases, and so does the execution time. This may improve the performance of the model by a small percentage, but the tradeoff is very high.

The Betweenness centrality range for the networks in consideration is shown in Table VI. The betweenness centrality measure denotes that how often a particular edge (say 'x') gets visited among the total paths in the network across any two nodes. This value, thus, will be in the range 0 to 1. Also, availability of such paths passing through edge 'x' in comparison to the total number of paths between any two nodes in the network will be very low. Hence, the betweenness centrality value evaluated for each edge as per explanation in subsection B of section 3, this value will be a very small number. However, the values can be normalized to any range/interval, but it will not affect the result as the magnitude of the betweenness centrality value increases for each edge by same factor.

TABLE VI. Citation Networks Accuracy

Network	**Minimum**	**Maximum**
CORA	0	0.0359
Citeseer	0	0.0462
PubMed	0	0.0134
Amazon	0	0.0055
WikiCS	0	0.0165

Bet-GCN performance over Facebook-Pages-Food Dataset: To further demonstrate the generalizability of Bet-GCN model in the perspective of social links of a social media platform, the model has been tested over Facebook-Pages-Food [43] network dataset.

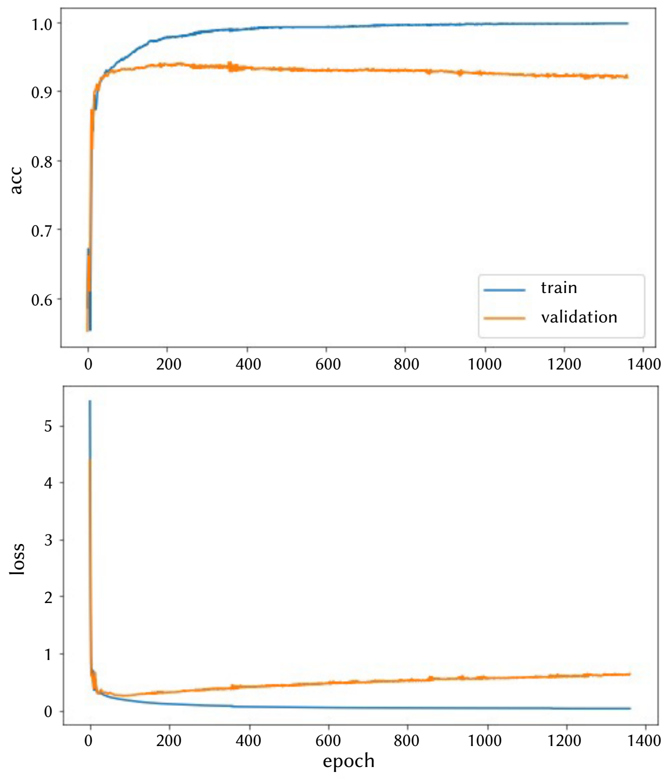


Fig. 20. Training and Loss Accuracy Curves for Facebook FoodWeb Pages.

Most social media platforms, including Facebook, can be structured as graphs. The registered users are interconnected in a universe of networks. The objective of link prediction is to identify pairs of nodes that will either form a link or not in the future. Here, we worked on a graph dataset in which the nodes are Facebook pages of popular food joints and well-renowned chefs from across the globe and if any two pages (nodes) like each other, then there is an edge (link) between them. For calculating node embeddings we have applied node2vec [44] on the graph. Then, Bet-GCN model is trained on 2259 edges and tested for 2522 edges. On training for 1500 epochs we get f1-score of 0.9442, which is a major improvement when compared to f1-score of 0.7817 for logistic regression in [43]. The model hyperparameter settings have been kept same as for the above models. The training and loss accuracy curves have been shown in Fig. 20 represents the training and loss accuracy curve for the same. This further demonstrates the prediction capability of Bet-GCN model with high accuracy on a different variety of real world graphical networks.

Reason of selection of GCN based methods over classical Machine learning and neural network techniques: The problem with social media data is the availability of the feature for every node in the graphical network. So, using correlational analysis and belief propagation techniques are not suitable as these techniques require features to be compared to calculate the similarity between the nodes and their behavior. In the absence of feature-based information, graphical structure information needs to be employed. The proposed high betweenness edge centrality based selection will lead to generation of computation graphs with more number of nodes (in average) during training. Since, feature set aggregation is directly proportional to number of nodes in the underlying computational graph, a better training of the GCN model is guaranteed using proposed approach. This in turn enhances the prediction capability of the model. In the state of the art literature there are many evidences where GCN based methods are outperforming traditional machine learning methods. Jiang et al. [45] have shown that the performance of GCN model to predict synergistic drug combinations in particular cancer cell lines in comparison to classical machine learning algorithms like Support Vector Machine, Radial Basis Function, Deep Neural Networks etc. is much better. Tayal et al. [46] have shown that the performance of GCN based techniques for text classification task is superior in comparison to other ML and DL techniques like TF-IDF with Logistic regression, CNN, Char CNN etc. The performance improvement is of approximately 2% with reduced dataset for training. Cao et al. [47] have shown in their comprehensive review article that how GCNs surpassed the performance of various CNN models. From these discussions, we can conclude that GCNs have high prediction capability due to added power of network structural information. Moreover, they can work well with limited feature availability and information about many data points in the network.

Lastly, lets have a look on the computational complexity of the model. The time complexity for calculating betweenness centrality of edges in the network is given as $O(|V|.|E|)$ [48], where, |E| are the number of edges and |V| are the number of nodes or vertices in the network. Further, the time complexity of GCN based training is given as $O(L.|V|.|F^2|)$ [49]. Here, 'L' represents the number of layers of the neural network, 'V' represents number of vertices and 'F' represent feature vector corresponding to each node of the graphical network. Then, overall complexity for the algorithm can be given as:

$$T = O(|V|.|E|) + O(L.|V|.|F^2|) \tag{26}$$

For real world networks, $|E| >> |V|$, but $|E| < |V|^2$. Therefore,

$$O(|V|^2) << O(|V| * |E|) < O(|V|^3) \tag{27}$$

Also, $L << |V|$ and is a constant value, so it can be omitted. Since, $F < |V|$, this means that $F^2 << V^2$. Therefore, from equations (26) and (27), we have,

$$O(|V|.L.|F^2|) \approx O(|V| * |F^2|) < O(|V|^3) \tag{28}$$

Hence, the overall time complexity of Bet-GCN model evaluates out to be of cubic order as a function of number of vertices. This means that solution is attainable in polynomial time. Moreover, the training process takes into consideration only 50 - 55% nodes into consideration. Given the advancements in computational power of modern day computers having GPU processors, the task can be accelerated significantly despite of cubic order time complexity of the process. Also, it is to be noted that even in case of traditional GCN the time complexity will be upper bounded by $O(L.|V|.F^2) \approx O(|V|^3)$. So, betweenness centrality based calculation do not hurt the overall time complexity of the task.

VI. Conclusions

The paper presents a variation of the traditional Graph Convolutional Network approach for the task of link prediction. An approach based on betweenness centrality was chosen for the selection of the edges to be trained. Thus, the top-k edges are selected to create the training set of edges that have a high value for edge centrality. This idea contributes to a significant improvement in model accuracy. The proposed model outperforms other state of the art based deep learning methods as the results are promising even with a high percentage of test dataset. The accuracy of the model was tested for up to 45% test dataset, while most state of the art models have reported accuracy over 5 - 10% test dataset. The reason for this improvement is the increased neighborhood span, which helps in generating rich node embeddings in GCN-based training for the model. The effectiveness of the results in the three datasets: CORA Citeseer and PubMed, was confirmed by the AUC curves. Moreover, the model has achieved impressive results on Amazon Product, WikiCS and Facebook Food

Web Page networks, which are very large and belong to a different category than the previous three, showing that the method is generic and can be applied to graphical networks of different domains. In summary, the key contributions of the manuscript are:

- Proposing an efficient GCN-based link prediction technique where the training set is selected based on *edge betweenness centrality*.
- Mathematical and experimental justifications of the improvement in GCN based training for link prediction.
- Detailed comparison of the results with the current state of the art methods for link prediction by performing experimental simulations over 6 different networks.

In future, the model can be tested with a larger number of complex network datasets to further verify the robustness of the proposed model. Moreover, the same model can be tested to determine the performance improvement on other tasks such as node classification, graph classification etc.

References

[1] M. Sun, J. Chen, Y. Tian, Y. Yan, "The impact of online reviews in the presence of customer returns," *International Journal of Production Economics*, vol. 232, p. 107929, 2021, doi: 10.1016/j.ijpe.2020.107929.

[2] M. S. Ullal, C. Spulbar, I. T. Hawaldar, V. Popescu, R. Birau, "The impact of online reviews on e-commerce sales in india: A case study," *Economic Research- Ekonomska Istraživanja*, vol. 34, no. 1, pp. 2408–2422, 2021, doi: 10.1080/1331677X.2020.1865179.

[3] M. Caro-Martínez, G. Jiménez-Díaz, J. A. Recio- García, "Local model-agnostic explanations for black- box recommender systems using interaction graphs and link prediction techniques," *International Journal of Interactive Multimedia and Artificial Intelligence*, pp. 1–11, 2021, doi: 10.9781/ijimai.2021.12.001.

[4] D. Medel, C. González-González, S. V. Aciar, "Social relations and methods in recommender systems: A systematic review," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 7, no. 4, pp. 7– 17, 2022, doi: 10.9781/ijimai.2021.12.004.

[5] N. N. Daud, S. H. Ab Hamid, M. Saadoon, F. Sahran, N. B. Anuar, "Applications of link prediction in social networks: A review," *Journal of Network and Computer Applications*, vol. 166, p. 102716, 2020, doi: 10.1016/j.jnca.2020.102716.

[6] S. Sledzieski, R. Singh, L. Cowen, B. Berger, "Sequence- based prediction of protein-protein interactions: a structure-aware interpretable deep learning model," *bioRxiv*, 2021, doi: 10.1016/j.cels.2021.08.010.

[7] M. Lim, A. Abdullah, N. Jhanjhi, M. K. Khan, M. Supramaniam, "Link prediction in time-evolving criminal network with deep reinforcement learning technique," *IEEE Access*, vol. 7, pp. 184797–184807, 2019, doi: 10.1109/ACCESS.2019.2958873.

[8] H. Huang, J. Tang, L. Liu, J. Luo, X. Fu, "Triadic closure pattern analysis and prediction in social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 12, pp. 3374–3389, 2015, doi: 10.1109/TKDE.2015.2453956.

[9] M. S. Granovetter, "The strength of weak ties," *American journal of sociology*, vol. 78, no. 6, pp. 1360– 1380, 1973.

[10] Y. Bi, W. Wu, L. Wang, "Community expansion in social network," in *International Conference on Database Systems for Advanced Applications*, 2013, pp. 41–55, Springer.

[11] E. Abbe, A. S. Bandeira, G. Hall, "Exact recovery in the stochastic block model," *IEEE Transactions on information theory*, vol. 62, no. 1, pp. 471–487, 2015, doi: 10.1109/TIT.2015.2490670.

[12] C. Matias, V. Miele, "Statistical clustering of temporal networks through a dynamic stochastic block model," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 79, no. 4, pp. 1119–1141, 2017.

[13] A. K. Gupta, N. Sardana, "Significance of clustering coefficient over jaccard index," in *The International Conference on Contemporary Computing*, 2015, pp. 463– 466, IEEE.

[14] D. Liben-Nowell, J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007, doi: 10.1145/956863.956972.

[15] S. Cohen, B. Kimelfeld, G. Koutrika, "A survey on proximity measures for social networks," in *Search computing*, 2012, pp. 191–206, Springer.

[16] S. Zhang, H. Tong, J. Xu, R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks*, vol. 6, no. 1, pp. 1–23, 2019, doi: 10.1186/s40649-019-0069-y.

[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020, doi: 10.1109/TNNLS.2020.2978386.

[18] T. Derr, Y. Ma, W. Fan, X. Liu, C. Aggarwal, J. Tang, "Epidemic graph convolutional network," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 160–168.

[19] H. Kashima, T. Kato, Y. Yamanishi, M. Sugiyama, K. Tsuda, "Link propagation: A fast semi-supervised learning algorithm for link prediction," in *Proceedings of the 2009 SIAM international conference on data mining*, 2009, pp. 1100–1111, SIAM.

[20] R. Raymond, H. Kashima, "Fast and scalable algorithms for semi-supervised link prediction on static and dynamic graphs," in *Joint european conference on machine learning and knowledge discovery in databases*, 2010, pp. 131–147, Springer.

[21] A. K. Menon, C. Elkan, "Link prediction via matrix factorization," in *Joint european conference on machine learning and knowledge discovery in databases*, 2011, pp. 437–452, Springer.

[22] S. Gao, L. Denoyer, P. Gallinari, "Temporal link prediction by integrating content and structure information," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, 2011, pp. 1169–1174.

[23] Z. Zeng, K.-J. Chen, S. Zhang, H. Zhang, "A link prediction approach using semi-supervised learning in dynamic networks," in *The International Conference on Advanced Computational Intelligence*, 2013, pp. 276–280, IEEE.

[24] L. Berton, J. Valverde-Rebaza, A. de Andrade Lopes, "Link prediction in graph construction for supervised and semi-supervised learning," in *The International Joint Conference on Neural Networks*, 2015, pp. 1–8, IEEE.

[25] T. N. Kipf, M. Welling, "Variational graph auto- encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[26] H. Yang, S. Pan, P. Zhang, L. Chen, D. Lian, C. Zhang, "Binarized attributed network embedding," in *IEEE International Conference on Data Mining*, 2018, pp. 1476– 1481, IEEE.

[27] P. V. Tran, "Multi-task graph autoencoders," *arXiv preprint arXiv:1811.02798*, 2018.

[28] R. Hisano, "Semi-supervised graph embedding approach to dynamic link prediction," in *International Workshop on Complex Networks*, 2018, pp. 109–121, Springer.

[29] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," *arXiv preprint arXiv:1802.04407*, 2018.

[30] X. Di, P. Yu, R. Bu, M. Sun, "Mutual information maximization in graph neural networks," in *The International Joint Conference on Neural Networks*, 2020, pp. 1–7, IEEE.

[31] T. Zhang, K. Zhang, X. Li, L. Lv, Q. Sun, "Semi- supervised link prediction based on non-negative matrix factorization for temporal networks," *Chaos, Solitons & Fractals*, vol. 145, p. 110769, 2021, doi: 10.1016/j.chaos.2021.110769.

[32] E. Bastami, A. Mahabadi, E. Taghizadeh, "A gravitation-based link prediction approach in social networks," *Swarm and evolutionary computation*, vol. 44, pp. 176–186, 2019, doi: 10.1016/j.swevo.2018.03.001.

[33] W. Gu, F. Gao, R. Li, J. Zhang, "Learning universal network representation via link prediction by graph convolutional neural network," *Journal of Social Computing*, vol. 2, no. 1, pp. 43–51, 2021, doi: 10.23919/JSC.2021.0001.

[34] M. Shabaz, U. Garg, "Predicting future diseases based on existing health status using link prediction," *World Journal of Engineering*, 2021, doi: 10.1108/WJE-10-2020- 0533.

[35] M. Wang, L. Qiu, X. Wang, "A survey on knowledge graph embeddings for link prediction," *Symmetry*, vol. 13, no. 3, p. 485, 2021, doi: 10.3390/sym13030485.

[36] F. J. Roethlisberger, W. J. Dickson, *Management and the worker*, vol. 5. Psychology press, 2003.

[37] R. Saxena, M. Jadeja, "Network centrality measures: role and importance

in social networks," in *Principles of Social Networking*, 2022, pp. 29–54, Springer.

[38] U. Brandes, S. P. Borgatti, L. C. Freeman, "Maintaining the duality of closeness and betweenness centrality," *Social Networks*, vol. 44, pp. 153–159, 2016, doi: 10.1016/j.socnet.2015.08.003.

[39] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann, "Pitfalls of graph neural network evaluation," *arXiv preprint arXiv:1811.05868*, 2018.

[40] P. Mernyei, C. Cangea, "Wiki-cs: A wikipedia-based benchmark for graph neural networks," *arXiv preprint arXiv:2007.02901*, 2020.

[41] Z. Zhang, X. Wang, W. Zhu, "Automated machine learning on graphs: A survey," *arXiv preprint arXiv:2103.00742*, 2021.

[42] M. Kaur, H. Kaur, "Implementation of enhanced graph layout algorithm for visualizing social network data using networkx library," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 3, 2017, doi: 10.26483/ijarcs.v8i3.2998.

[43] R. Rossi, N. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

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

[45] P. Jiang, S. Huang, Z. Fu, Z. Sun, T. M. Lakowski, P. Hu, "Deep graph embedding for prioritizing synergistic anticancer drug combinations," *Computational and structural biotechnology journal*, vol. 18, pp. 427–438, 2020, doi: 10.1016/j.csbj.2020.02.006.

[46] K. Tayal, R. Nikhil, S. Agarwal, K. Subbian, "Short text classification using graph convolutional network," in *NIPS workshop on Graph Representation Learning*, 2019.

[47] P. Cao, Z. Zhu, Z. Wang, Y. Zhu, Q. Niu, "Applications of graph convolutional networks in computer vision," *Neural Computing and Applications*, pp. 1–19, 2022, doi: 10.1007/s00521-022-07368-1.

[48] N. Kourtellis, G. D. F. Morales, F. Bonchi, "Scalable online betweenness centrality in evolving graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2494–2506, 2015, doi: 10.1109/ICDE.2016.7498421.

[49] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 257–266.

Rahul Saxena

He is currently working as an Assistant Professor in Department of Information technology, Manipal University Jaipur, since 2015 and pursuing PhD from Malaviya National Institute of Technology, Jaipur since 2019. He completed his Masters from Manipal University Jaipur in the year 2015. He has been awarded with Gold Medal for Excellence in Education in Masters. He completed his B.E. from Birla Institute of Technology, Mesra in year 2013. His areas of research and interest includes Social Networks Analysis, Machine Learning, Graph Algorithms, Parallel processing etc. He has several conference, journal articles and book chapters published in Springer, IEEE etc. in the related domains of research.

Spandan Pankaj Patil

She received B.tech in Electrical engineering degree in 2022 from NIT Jaipur. She is currently working as a full-time Data Scientist at Micron Technology. Her research interests include Graph neural networks, social network analysis, machine learning, and computer vision.

Pranshu Vyas

He received his B. Tech. in computer science and engineering from MNIT Jaipur. He is currently working as a software developer at D. E. Shaw India Private Limited. His fields of interest are Data structure, Algorithms, and Machine learning, with a special focus on Neural network approaches for graphical data such as GCN.

Atul Kumar Verma

He received his B.Tech degree in computer science and engineering from VBS Purvanchal University, Jaunpur, UP, India in 2009 and M.Tech degree in computer science and engineering from Dr. A.P.J. Abdul Kalam Technical University UP, India in 2016. He is currently pursuing PhD at the Department of computer science and engineering, Malaviya National Institute of Technology, Jaipur India. His areas of interest are Social Networks Analysis, Machine Learning, Deep Learning and Graph Algorithms.

Mahipal Jadeja

He received his Ph.D. degree from Dhirubhai Ambani Institute of Information and Communication Technology (DA-IICT) in the field of Theoretical Computer Science. He currently works at Malaviya National Institute of Technology (MNIT Jaipur) as an assistant professor. His research interests include Theoretical Computer Science, Social Network Analysis, and Machine Learning on Graphs (Graph Neural Networks). He has published several journal articles, book chapters, and a reference book (Springer) in these domains. His research work is presented at reputed international conferences including GSB-SIGIR 2015 (Chile), WAAC 2016 (Japan), and SCAI-ICTIR 2017 (Netherlands).

Vikrant Bhateja

Vikrant Bhateja is associate professor in Department of Electronics Engineering Faculty of Engineering and Technology, Veer Bahadur Singh Purvanchal University, Jaunpur, Uttar Pradesh, India. He holds a doctorate in ECE (Bio-Medical Imaging) with a total academic teaching experience of 19+ years with around 190 publications in reputed international conferences, journals and online book chapter contributions; out of which 35 papers are published in SCIE indexed high impact factored journals. Among the international conference publications, four papers have received "Best Paper Award ". Among the SCIE publications, one paper published in Review of Scientific Instruments (RSI) Journal (under American International Publishers) has been selected as "Editor Choice Paper of the Issue" in 2016. He has been instrumental in chairing/co-chairing around 30 international conferences in India and abroad as Publication/TPC chair and edited 50 book volumes from Springer-Nature as a corresponding/co-editor/author on date. He has delivered nearly 20 keynotes, invited talks in international conferences, ATAL, TEQIP and other AICTE sponsored FDPs and STTPs. He has been Editor-in-Chief of IGI Global--International Journal of Natural Computing and Research (IJNCR) an ACM & DBLP indexed journal from 2017-22. He has guest edited Special Issues in reputed SCIE indexed journals under Springer-Nature and Elsevier. He is Senior Member of IEEE and Life Member of CSI.

Jerry Chun-Wei Lin

He received his Ph.D. from the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan in 2010. He is currently a full Professor with the Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway. He has published more than 500+ research articles in refereed journals (with 90+ ACM/IEEE transactions journals) and international conferences (IEEE ICDE, IEEE ICDM, PKDD, PAKDD), 16 edited books, as well as 33 patents (held and filed, 3 US patents). His research interests include data mining and analytics, natural language processing (NLP), soft computing, IoTs, bioinformatics, artificial intelligence/machine learning, and privacy preserving and security technologies. He is the Fellow of IET (FIET), ACM Distinguished Member (Scientist), and IEEE Senior Member.