Prediction of Protein Protein interactions using Graph Convolutional Networks

Akshat Sharma
sharma06@ads.uni-passau.de
Universität Passau

Amit Manbansh
manban01@ads.uni-passau.de
Universität Passau

Lovesh Bishnoi
bishno01@ads.uni-passau.de
Universität Passau

Mihir Shah
shah02@ads.uni-passau.de
Universität Passau

1 INTRODUCTION[AKSHAT SHARMA]

In the body of all living creatures, proteins are building blocks of all cells. Some specialized cells in the human body bind with each other to identify what is new for the body. Such an identification process is possible via protein-protein interaction on the surface of the immune system cells. Thus, the affinity of binding cells can determine whether or not the immune process will be initiated. Protein and its interaction with other proteins are remarkably very complex phenomena. Knowledge of protein interaction can help pharmacists and microbiologists understand the function and behaviour of protein, to assign a new function. Adding to this, a cluster of proteins with the same function can be generated. Biologists and pharmacists can study protein interaction so that they can characterize protein complexes [3]. Study of protein-protein interaction can give answers to many questions of life on this planet, such as how one gets cancer, how one grows old, why one gets affected to disease, and how an antibody to a particular disease can be developed. These are the very few examples of protein functions, and that is why it is very important to study protein, its composition, its structure, its interaction with other proteins. Conventional approaches available to study protein protein interaction are bit time consuming and as well as expensive. Moreover, large scale experiments always involve high rate of false positive. On the other hand, computational algorithms developed due to recent developments in Machine Learning, can prove to be handy in identifying many undiscovered PPI's, that require high throughput experiments. Furthermore, such computationally discovered PPI's can be verified by human's and thus it can save time as well as human labour.

1.1 Graph Convolution Network[Akshat Sharma, Amit Manbash]

Graph Convolution Networks (GCNs) [8] are the neural network models that share the filter parameters over all locations in the graph. The goal of these neural network models is to learn a function of features on a graph. Let $G = (V, E)$ be the graph where V is the set of the Nodes and E is the set of the Edges. The inputs to the Graph Convolution Network are a feature description x_i for every node i, where $x_i \in X^{N \times D}$ and X is the feature matrix and N is the number of the nodes and D is the number of the input features, and A is the Adjacency Matrix which represents the description of the graph structure in matrix form. The Graph Convolution Network, i.e., G produces a node level output $Z^{N \times F}$, where F is the output features per node.

Every Graphical Neural Network layer can be represented mathematically as a non linear function [8]

$$H^{(l+1)} = f(H^{(l)}, A). \tag{1}$$

where $H^{(0)} = X$ and $H^{(l)} = Z$ and $'l'$ is the number of layers and $'f'$ is the function chosen for a specific task.
A simple layer wise Propagation rule [8] is given by

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)}) \tag{2}$$

where $W^{(l)}$ is a weight matrix for the l-th neural network and $\sigma(.)$ is a non linear activation function which could be ReLU, softmax, etc depending on the choice. Mathematically, these functions are defined as respectively:

$$ReLU(x) = \begin{cases} 0 & \text{if x} < 0 \\ \text{x} & \text{if x} \geq 0\ , \\ \end{cases} \tag{3}$$

$$softmax = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}. \tag{4}$$

There are two main limitations of this model and their possible adjustments are:

(1) Directly multiplication with A means that for every node all the feature vectors of all the neighbouring nodes are summed but not of that node under consideration. This limitation is overcome by adding the Identity Matrix, i.e., I [8] to the Adjacency matrix, i.e., A, mathematically:

$$\tilde{A} = A + I \tag{5}$$

where $\tilde{A}$ is the Adjacency Matrix with self loop and it is done so that each node also includes its own features at its next representation and it also helps with the numerical stability.

(2) A is not normalised which can lead to the change of scale of the feature vectors. This limitation is adjusted by normalising A such that all rows sum to one, i.e., $D^{-1}A$, where D is the diagonal node degree matrix of A and $D^{-1}A$ denotes the averaging of the neighbouring node features. Practically averaging of the neighbouring node features is not sufficient therefore symmetric normalisation or spectral approximation of A is done, i.e., $D^{\frac{-1}{2}} A D^{\frac{-1}{2}}$ [8].

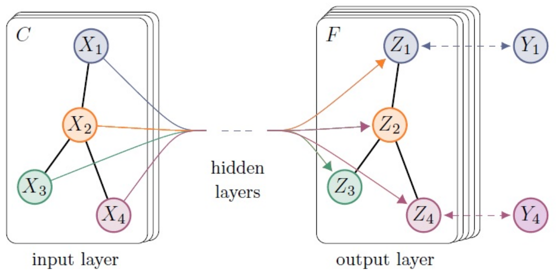

Figure 1: Graph Convolutional Network [8]

Combining these two adjustments to the above limitations we get a propagation rule [8]:

$$f(H^{(l)}, A) = \sigma(\tilde{D}^{\frac{-1}{2}}\tilde{A}\tilde{D}^{\frac{-1}{2}}H^{(l)}W^{(l)}) \tag{6}$$

Where, $\tilde{D}$ is the diagonal node degree matrix of $\tilde{A}$ and it is used to normalise the nodes with large degrees. A multi layer Graph Convolutional Network can be shown with C input channels and F feature maps in the output layer as shown in Figure 1.

In the research paper "Semi-Supervised Classification with Graph Convolution Networks" by Thomas Kipf et al [8], the authors tried to solve the problem of "Citation Networks" by using the the above mentioned techniques of the GCN propagation in a two layer GCN model for semi supervised node classification. Their model was mathematically represented as:

$$Z = f(X, A) = softmax(\tilde{A}ReLU(\tilde{A}XW^{(0)})W^{(1)}) \tag{7}$$

In the citation network the documents were considered nodes and edges were the documents cited in the published documents and on this semi-supervised classification the loss (cross entropy loss) was computed as:

$$L = -\sum_{l \in Y_L}\sum_{f=1}^{F} Y_{lf} \ln Z_{lf} \tag{8}$$

Where Y_L is the set of node indices with labels and F are the feature maps in the output layer

1.2 Protein-Protein Interaction[Akshat Sharma, Mihir Shah]

A protein is a large organic bio-molecule [2] which is formed from the combination of Amino Acids which combine with each other using a peptide bond [2] to create a protein molecule. The length of the protein depends on its genetic code which specifies the number and types of amino acids responsible in the formation of a particular protein. The genetic expression in any organism is directly related to the proteins associated with that genetic code. When these proteins interact with each other then this interaction is known as Protein-Protein Interaction (PPIs) [7]. Cellular functions are defined by the interaction between the proteins, therefore if we want to understand the basic cell functions, we need to map the protein-protein interactions. Protein interactions have a huge potential within an organism as a whole, e.g., total interactions of human PPIs [4] are estimated to be around 650,000.

There are mainly two methods to detect the protein-protein interactions [4], namely Experimental and Computational. Experimental methods include Y2H, TAP-MS, protein microarrays, mbSUS, pull-down arrays, DPI, etc. Computational methods include PTMs, gene fusion, co-expression, GO annotation, gene neighbourhood, Phylogenetic profile, topological features, sequence features, Domain interactions, protein fold, etc.

Computational techniques consider "protein–protein interactions [7]" or PPIs as the associations or links between proteins. These techniques overcome the limitations of experimental techniques, e.g., Experimental findings are often incomplete even for well-studied organisms, therefore computational methods are used to complete the missing or incomplete part of the experimental PPI data and thus help in finding the clues to map out PPI mechanisms. These methods mainly focus on individual evidence for prediction and have certain specificities and biases. The various evidence sources are integrated in a statistical learning framework, such methods are called "prediction of protein-protein interactions by evidence-combining methods". The machine learning techniques can be applied on such a statistical framework.
The machine learning algorithm on the prediction of protein-protein interactions by evidence-combining methods mainly consists of three steps:

(1) Defining Gold Standard Datasets [4] (training datasets of interacting and non-interacting protein pairs).
(2) Characterising the interactions between proteins by annotating the Gold Standard Datasets [4] with carefully chosen and diverse evidence.
(3) Determining the probability of a particular interaction or interactions by individual evidence and then combining the probabilities of all the evidences.

1.2.1 Gold Standard Datasets. They are created for training or testing the PPI predictions and the datasets for training and testing are separate. These datasets could be either GSP datasets (Gold Standard Positive) or GSN datasets (Gold Standard Negative).

GSPs are PPIs with high experimental confidence or reliability or reference evidence, e.g., BioGRID, IntAct, etc, are some examples which are available in public databases. They are mainly the repositories of protein complexes and interactions are varied in terms of size and species-specificity and they contain information from both the experimental and the computational sources.

GSNs are usually not obtained by direct experimental methods or techniques. Negatome Database (2.0) provides a collection of proteins and domain pairs which are unlikely to engage in direct interaction but it wasn't able to satisfy the diverse GSP datasets of different users. GSNs can be obtained using certain methods, e.g., Negative examples can be chosen from the categories of their particular functions like annotations and subcellular localisation, etc.

1.2.2 Annotations of protein pairs with diverse evidence. Protein pairs are annotated based on their interactions with each other which could be based on cell physiology, biochemical environment, structures of the protein complexes, etc. To detect PPIs experimentally, certain conditions and criteria are met depending on the nature of protein interaction. For prediction of the PPIs by machine learning algorithms, we need to extract the protein interaction-based features. As there are several conditions for different PPIs therefore features are categorised into different categories and each category can provide a different view of protein interactions. Some of these categories and the features contained in them are: Evolutionary relationship (EVO) with features related to genes, e.g., Gene Neighbourhood (GN) etc, Functional Features (FF) and Sequence based code signatures (SEQ), Structure based signatures which are based on the structural interactions of the proteins and the features included in this category are Domain Domain Interaction (DDI), etc. DDI is represented in the Figure 2

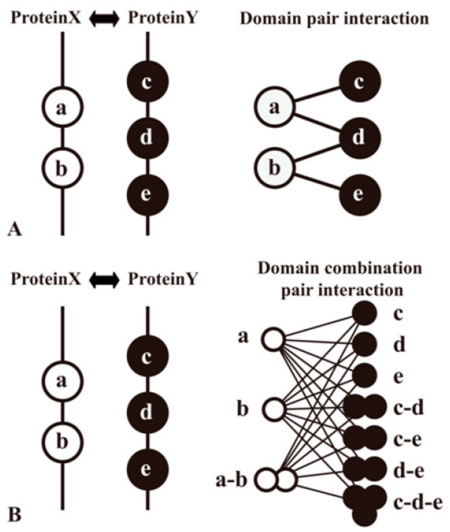

Figure 2: Types of Domain Domain Interactions [4]

Figure 2 shows two methods to predict Domain Domain Interactions from PPIs. Consider two proteins with domains {a, b} and {c, d, e} respectively, here PPIs are interpreted as the interaction amongst the domains of the two proteins.
In method A one domain of a protein will interact with one domain of the other protein.
In method B one or more domains of a protein will interact with one or more domains of the other protein.

1.2.3 *Strategy for Integrative Analysis.*

Classification Algorithm is used to integrate the protein interaction related features, with these available features, classifiers are trained to differentiate between positive and negative examples. The usual process of PPI prediction by evidence-combining techniques [4] includes mainly three steps which are:

(1) Step I Choose appropriate evidence.
(2) Step II Encode protein pairs with evidence.
(3) Step III Find strategy to merge the classifiers into the integrative datasets.

The strategies could be the use of Artificial Neural Networks, Naïve Bayes, Decision Tree, K Nearest Neighbours, Support Vector Machine, et cetera.

1.2.4 *Performance evaluation of PPI Prediction .*

The following techniques are considered to perform evaluation: Precision, Recall (Sensitivity or True Positive Rate), Specificity (True Negative Rate), Area Under the ROC curve, etc. They are defined using the terminology of the confusion matrix such as (True Positive)TP, (True Negative)TN, (False Positive)FP and (False negative)FN.

- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $Specificity = \frac{TN}{FP+TN}$

The ROC (Receiver operating characteristic) curve is shown in the following Figure 3[1].

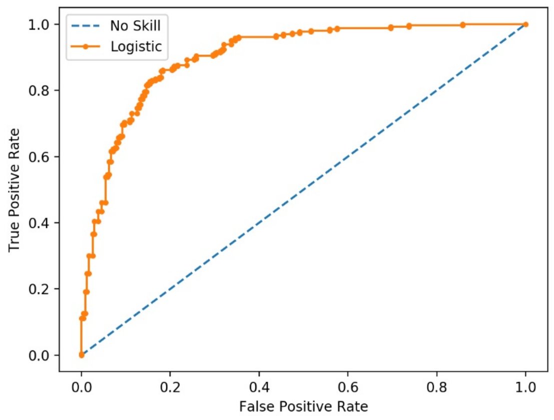

Figure 3: ROC Curve for a No Skill Classifier and a Logistic Regression Model

And area under the ROC curve is calculated using integration by considering the FPR (False Positive Rate) as the x-axis and the TPR (True Positive Rate) as the y-axis:

$$A = \int_{x=0}^{x=1} TPR(FPR^{-1}(x))dx \tag{9}$$

1.2.5 *PPI Prediction through Domain Domain Interactions.*

A domain is mainly one or more submolecular parts of protein, described as a structural and functional module and usually an evolutionarily conserved unit. Recent studies[9] about domains have concluded that abnormalities of domains can cause various diseases. Therefore, studies on the protein domain can help in developing disease models and tools to diagnose them. However, experimental techniques of the domain-domain interaction for predicting PPI have often shown more false-positive and false-negative results. The researchers have started studies on PPI prediction using computational techniques, that are mainly based on different features of protein such as protein sequence, domain information, three-dimensional structure, and protein evolution. The current state of the art approach does not fully consider domain information, instead only works on important domain and domain co-occurrences. However, an overall view of the PPI network can be better understood by domain-domain interaction [9].

[1]https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

1.3 Related work$_{\textbf{[Lovesh Bishnoi]}}$

Our model takes inspiration from the conventional computation approach for PPI prediction and recent development on the graph convolution network for PPI. In what follows, we provide a brief description of related work in both the fields.

1.3.1 Conventional Approach for PPI. In the paper "Prediction of Protein-Protein interaction based on Domain" by Xue Li. et.al [9] proposed a novel approach based on protein domain for Protein-Protein Interaction. In this approach, the authors have used a state-of-the-art SVM model, in which physochemical properties of domain and domain-domain interaction score are used as the features for the prediction model. The outcome of the SVM model and the domain-domain score were used to design a Protein interaction prediction model. Accuracy, sensitivity, specificity, precision, Matthews correlation coefficient, and the F1 score are used as the evaluation matrix of the prediction model. The drawback of this approach was that it was implemented on a very small scale of the dataset. Adding to this, for the unavailability of negative domain-domain pair data, the general noise was added instead.

1.3.2 Graph Convolution Network for PPI prediction . The spatial Graph convolution approach was used for protein interface prediction by Alex Fout et.al. [6] in the paper "Protein Interface Prediction using Graph Convolutional Networks". The prediction of the protein interface was based on the graph structure of the protein where amino acid residues are nodes. A set of k residues determined by mean distance between the atoms is used as a neighborhood-based convolution, which is able to detect the node features accurately. The edge features were also derived by their network, but in limited amount and were static compared to the node feature. Thus, the model learned the latent pattern for node features only. For evaluation, the authors compared their approach with State-of-the-art SVM based protein prediction approach. The AUC score for novel approach is 0.89 whereas, AUC score for the SVM model was 0.81. The authors suggest that their approach can be improved by using a large set of protein dataset for better representation for CNN and along with node feature, the model can be improved to learn edge feature representation.

1.4 Problem Definition$_{\textbf{[Akshat Sharma]}}$

In this project, we are trying to solve the problem of the time-consuming experimental process to find out the possibility of PPI for given proteins with a faster computational method. We will be using GCN to design a model which can predict the possibility of a link between two given proteins.

Mathematically, the problem can be summarized as: For a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, given two nodes $x, y \subseteq V$, where V represents proteins, we will implement a Graphical Neural Network model to predict if
Edge {x, y} $\subseteq E$ | Edge {x, y} $\not\subset E$.

1.4.1 ***Research Questions:*** We formulate our research questions as follows:

(1) How does increase in hidden-layers in GCN impact the accuracy in a link prediction task?
(2) What is the impact of various hyperparameters on the performance of GCN for a link prediction task?

2 DATA ACQUISITION & PRE-PROCESSING$_{\textbf{[LOVESH BISHNOI]}}$

Yeasts are the single-celled eukaryotic fungi microorganisms and they are unique because in the kingdom of Fungi as they are the only single-celled microbes whilst all other members are multicellular organisms. Here we are analysing the protein dataset of a particular yeast of Class: Saccharomycetes, Specie: S. cerevisiae and Genus: Saccharomyces.

2.1 Data acquisition$_{\textbf{[Akshat Sharma, Lovesh Bishnoi]}}$

We are using the protein dataset[2] of yeast provided by the Deep Learning for Network Biology[3] by the Stanford University. The Edgelist contains the pairs of proteins which interact with each other. The yeast Edgelist is a clean and a well organised dataset. To get a general idea about our yeast dataset or the Edgelist we read the Paper by Xiaomei Wu et al [10], in which a protein protein interactions (PPI) map was derived based on Gene Ontology (GO) annotations. It was done by measuring the similarity of two Gene Ontology terms with a semantic relation known as Relative Specificity Similarity (RSS) and using Z score analysis a positive and a negative datasets for PPI were created. A gold standard positive dataset (GSP) and a gold standard negative dataset (GSN) were also created with high levels of confidence (about 78 percent high quality) and low levels of confidence respectively.

Gene Ontology (GO) is a resource that has relative data sources integrated, which represents the knowledge of the genes in genomes, which give the particular information about certain biological roles attributed to the units associated with those genes, e.g., proteins. GO has been used in protein classification for many species, e.g., Saccharomyces cerevisiae, homo sapiens etc and has a set of three controlled vocabularies or structural ontologies , e.g., Molecular Function (MF), Biological Process (BP) and Cellular Component (CC).

Functional protein associations can be described by their shared GO terms in a structural ontology or by the semantic similarity of the protein pairs of the terms assigned to them through information or GO. Xiaomei Wu et al [10] worked on to predict a map of the yeast's PPIs by using the BP and CC ontologies or annotations. They followed mainly two particular assumptions which are:

(1) Two proteins which mostly function in the same biological process are more likely to interact with each other than the two proteins which function in different biological processes.
(2) For two proteins to interact they must be in a close proximity to each other.

Xiaomei Wu et al [10] used the Organelle DB which is an online resource used for the proteins in eukaryotic organisms localised to the organelles of the cell or in layman's language the subcellular structures. The proteins taken from the Organelle DB were annotated using the BP and CC ontologies from GO consortium by Xiaomei Wu et al [10].

[2]http://snap.stanford.edu/deepnetbio-ismb/ipynb/yeast.edgelist
[3]http://snap.stanford.edu/deepnetbio-ismb/

2.2 Data preprocessing[Amit Manbansh, Mihir Shah]

The chosen dataset is present in Edge List[4] format. A section of Yeast dataset edge list format file can be seen in Figure 4. Each line in this file represents an edge. The first string in each line represents a node from where the edge is directed. The second string of each line represents the node where the edge is directed to. The nodes here represent complex proteins. An edge between two nodes denotes that both proteins can interact with each other. For example, from line 1 of Figure , we can conclude that protein “YNL236W” can interact with protein “YGL238W”.

```
YNL236W YGL238W
YNL236W YOR355W
YNL236W YJL030W
YNL236W YJL013C
YNL236W YJR034W
YNL236W YKL012W
YNL236W YFR033C
YNL236W YGR046W
YNL236W YGR117C
YGL208W YGL115W
YDR328C YLR399C
YDR328C YFL009W
YDR328C YMR094W
YDR328C YJR090C
YDR328C YIL046W
YDR328C YDR139C
YDR328C YOR057W
```

Figure 4: Edgelist

2.2.1 ***Statistics on the data:*** Networkx [1] library has been used to read the Yeast Edge List format file. Upon reading the Edge List format file into undirected graph[5] format and converting it to Compressed Sparse Row matrix[6] format, we find that there are 6526 nodes in the network. Also, there are 1062675 edges in the Compressed Sparse Row matrix[7].

2.2.2 ***Handling missing data:*** In Figure 5, we can see node “-” is 11th most centralized node. Since “-” does not represent any protein in the real world, this missing data needs to be removed from the adjacency matrix. The node “-” is connected to 1291 proteins and it is removed from the adjacency matrix resulting in deletion of 2582(1291 * 2) edges from the adjacency matrix. The resulting remaining nodes are 6525 and remaining edges are 1060093.

2.2.3 ***Handling self-loop:*** Upon checking further we find that the matrix contains weight 1 inside 1685 elements which are present in diagonal of the matrix. Upon deletion of 1685 edges present at diagonal, there are 1058408 edges remaining in the graph. Since, this represents that A protein is can interact with itself, it doesn’t carry any meaning of any significance in this project. We removed such elements from the diagonal of the matrix.

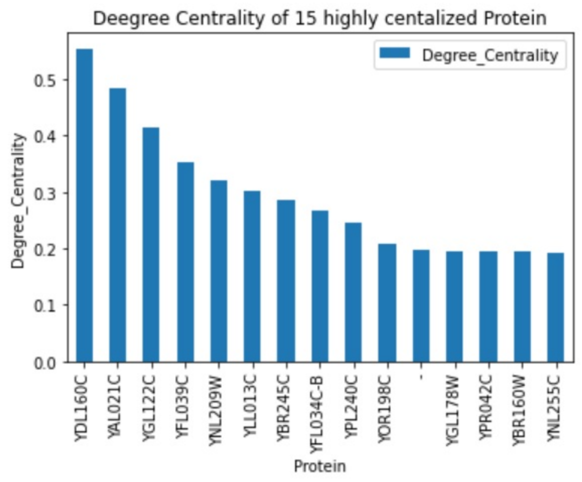

Figure 5: Degree Centrality of 15 highly centralized Proteins

2.2.4 ***Handling directed and symmetrical behaviour in Adjanceny matrix:*** Since Yeast PPI is an undirected activity, we get rid of symmetrical data of Adjanceny matrix by considering only the Upper Triangle of the matrix. We noticed that element[A, B] and element[B, A] contains the same value in the matrix. It represents that Protein A can interact with Protein B and Protein B can interact with Protein A. For example, the element at index [0,1801] and [1801, 0] has the value 1. Index 0 represents YAL008W protein and Index 1801 represents YNR020C protein. Both the cell denotes the same data and hence, we will be considering only the Upper Triangle matrix to extract interaction edges data for the training, validation and testing purpose.
After considering upper Triangle elements of the matrix, we are left with 529204 edges from the total 1060990.

2.2.5 ***Degree Centrality.*** After understanding some basic structure measures of the whole network, a good next step is identify the most important nodes in the network. In graph or network, number of edge connection of a particular node to other nodes in the network is measured by degree. Degree of a network or graph can tell about the biggest hubs, while it can not give more understanding for nodes. Adding to this, in network analysis, identifying the most important nodes in the network is measured by centrality. Moreover, the dictionaries that gives nodes as keys and centrality measures as values is called degree centrality. Thus, we can identify with numeric value, the importance of particular node in the network. Degree Centrality of some of the proteins in the dataset are shown in the Figure 5.

2.2.6 ***Eigenvector Centrality.*** After getting through the basic visualization of the network, a good next step is to identify which nodes are most important in the network, which nodes form communities/groups within the network.

In graph network, analysis of most important in the network is referred to as the degree of centrality. One of the metrics to find the

[4]http://snap.stanford.edu/deepnetbio-ismb/ipynb/yeast.edgelist
[5]https://networkx.github.io/documentation/networkx-1.9.1/_modules/networkx/classes/graph.html
[6]https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html
[7]https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html

most important node is a degree, which is considered as the most simple metric. The degree of a node is the number of edges extend from it. For example, a degree node with degree 3 is defined as 3 edges extending from it to other nodes in the network. Another metric for centrality measurement is eigenvector centrality, which is an extension of degree centrality. Eigenvector centrality not only looks at the edges of a particular node but it also takes into account the edges of its neighboring nodes. Centrality in PIP interaction can prove to be handy to identify which protein interacts to the most with other proteins in the network. Then it can later be helpful to study the other features of proteins interacting. In yeast dataset, Protein YDL160C has a degree centrality of 0.5546360153256705, and eigenvector centrality of 0.10615358103813279 which is highest in the network as shown in the Figure 6

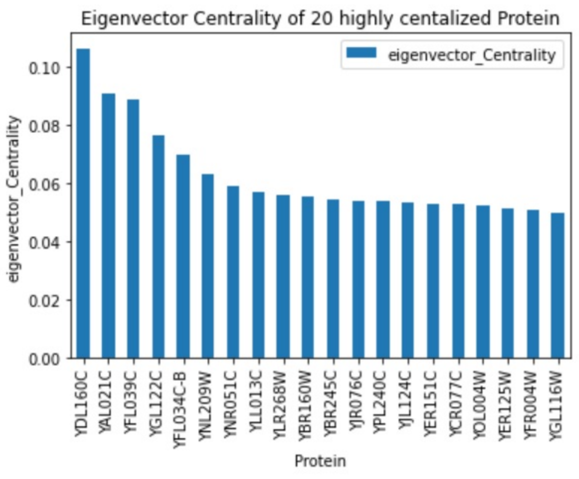

Figure 6: Eigenvector Centrality of 20 highly centralized proteins

2.2.7 ***Community detection***.

Community detection is the identification of communities or groups in a large sparse network. One such community detection technique is greed_modularity_communities(), and groups the nodes into subsets according to the connected community. There is one set for each group and it contains nodes present in that community. In yeast dataset 7 such community of interacting proteins are present (condition: len_community >2). Eigenvector centrality can be used in community detection to find, the most important node in particular group.

2.3 Feature Engineering[Amit Manbansh, Lovesh Bishnoi]

2.3.1 ***Training/Validation/Testing set size:***

We aim to check the performance of the designed model on different size of training data. We plan to vary the training set size from 60% to 96% of the total data available.

2.3.2 ***Method of construction of Testing/Validation/Testing set:***

Suppose we are aiming to check the performance of the designed model on x% of all available data as the training set. Therefore, We pick (100-x)/2% of total edges randomly each for the validation as well as testing purpose respectively. The remaining edges are used for constructing the training edge list. We also created the same number of False edge list each for validation as well as testing purpose respectively.
For choosing Element[A, B] is eligible for being a part of the Validation false edge list, we checked if Element[A, B] satisfies 5 checks. If all the criteria are satisfied, we added the Element[A, B] to the validation false edge list. The rules are as follows:

(1) Check whether Node A and Bode B are not the same.
(2) Check if Element[A, B] is not a member of the training edge list.
(3) Checked if Element[B, A] is not a member of the training edge list.
(4) Check if Element[A, B] is not a member of the validation edge list.
(5) Checked if Element[B, A] is not a member of the validation edge list.

For choosing Element[A, B] is eligible for being a part of the testing false edge list, the checks were a little different from the checks for choosing validation false edge list. We just checked if Element [A, B] is a member of all the edges present in Upper Triangle of the Adjacency matrix or a diagonal element. If not, Element [A, B] was added to testing false edge list. The reason is to sample only those edges for the testing set which are unseen to the model.
Feature transformation of training adjacency matrix: We also normalized the Adjacency matrix created from the training edge list. We converted it into symmetric normalization, i.e., $D^{\frac{-1}{2}}AD^{\frac{-1}{2}}$ as suggested in [8]. We can see the count of different edge list in the following Table 1

Table 1: Various counts from the Yeast Edgelist.

Total Nodes	6526
Total Edges	1062675
Total Nodes after removing invalid node "-"	6525
Total Edges after removing invalid node "-"	1060093
Total Edges after removing diagonal elements	1058408
Total edges in Upper Triangle of Adjacency Matrix	529204

3 MODEL IMPLEMENTATION[MIHIR SHAH]

3.1 Methodology[Akshat Sharma]

3.1.1 *Build a Graph Convolution Network.*

In our proposed solution we use the PyTorch Geometric[5] PyG, which is a geometric deep learning extension library for PyTorch, we use it because of its various features like, an easy-to-use mini-batch loader for many small and single giant graphs and multi GPU-support. The GCN layers are already defined in the PyG library as method and mathematically the Graph Convolution Operator[8] is defined as

$$\mathbf{X}' = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Theta} \tag{10}$$

Where, $\mathbf{\Theta}$ is the weight matrix, and $\hat{\mathbf{A}}$ is the adjacency matrix with self loops from Equation (5) and $\hat{D}_{ii} = \sum_{j=0} \hat{A}_{ij}$ is the its diagonal degree matrix. the GCN layer in PyG[5] is defined as

$$\mathbf{x}_i^{(k)} = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{\deg(i)} \cdot \sqrt{\deg(j)}} \cdot \left(\mathbf{\Theta} \cdot \mathbf{x}_j^{(k-1)} \right) \tag{11}$$

The Equation (11) can be explained in six steps, which are, adding self loops to the adjacency matrix, Linearly transforming node feature matrix, Computing normalisation coefficients, Normalising node features in the activation function of the first GCN layer, Summing up neighbouring node features, Returning new node embeddings in the activation function in the next layer.

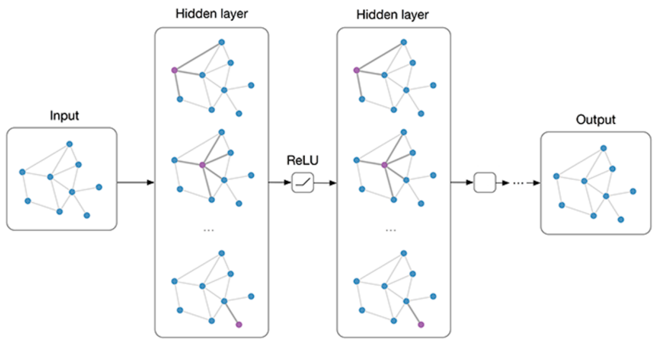

Figure 7: Multi-layer Graph Convolutional Network (GCN)

The inbuilt method of PyG to represent the GCN layer can be used as a building block for deep neural architectures, i.e., several layers can be defined using this function as shown in the Figure 7[8] approximately. In our solution, we have defined two layers, the first layer consists of 6526 input features and 32 output features and the second layer consists of 32 input features and 16 output features. We define the forward propagation of these layers as stated in equation (1), In the first layer, we use the ReLU activation function as described in Equation (3) and in the second layer, we give the output of the first layer as the input, which is the adjacency matrix with self-loops as stated in Equation (5) for positive train. Further, we use the inner product decoder for link prediction.

3.1.2 Training. The model training is initialised to the train mode. The training node feature data is labeled as positive edge index tensors. The negative sampled data is generated by adding self loops to the positive labeled data, which is further given as input to the negative sampling function method. In the further step, the model is called with positive edge index and negative edge index as arguments, which means the GCN is executed layer wise, the first layer computes the product of the normalised adjacency matrix and the training positive edge index and this product is passed through the ReLU activation function and then output of this layer is fed as an input to the second layer which computes the product of it and the training positive edge index matrix. Now the positive edge index and the negative edge index are concatenated and then taking the reference of the rows of the concatenated total edge index as indices two matrices are formed from the output of the second layer. We used row wise dot product on these two matrices which is mathematically represented as $c_i = \sum_j A_{ij} B_{ij}$. We also get the labels from the positive and the negative edge indices and finally we calculate the loss, i.e., the binary cross entropy which will pass the output of the GCN through the sigmoid function and lastly the backpropagation is done and it will return the loss calculated above. The model uses Adam optimiser with learning rate 0.01. The model is trained for 1 to 20 epochs.

3.2 Experimental Setup[Mihir Shah]

In order to achieve PPI interaction for the Yeast S. Cerevisiae dataset, GCN deep learning model is built in PyTorch (version 1.5.0). Pytorch has set its benchmark in the field of computer vision and natural language processing. It is an open-source machine learning framework. PyTorch is considered to be one of the simplest deep learning frameworks to learn because most of its syntax and applications are similar to conventional deep learning libraries. PyTorch has a very unique feature caller data parallelization. Using this feature of PyTorch, one can distribute the computational work among several CPU or GPU cores. PyTorch stands out from other deep learning frameworks because PyTorch supports a dynamic computational graph. Dynamic computation of the graph is helpful to change the behavior of the network programmatically during run time. Thus, the dynamic approach of PyTorch helps to understand each and every computation in the network, which makes it first priority for researchers. Many tools and libraries are developed by the PyTorch community to support its dynamic nature. One such PyTorch extension to process geometric and irregular structured input data is PyTorch Geometric (PyG). In comparison with other network graph library, PyG showed very fast computing despite being sparse data.

3.2.1 Dataset Construction. Yeast dataset is a dense graph form dataset with nodes as proteins and interacting proteins connected as edges. Therefore, graph form data needs to be transformed using networkx library in python. Networkx[1] library provides a method to read edgelist, we used that method to read edges from network as a result we got edges connected in the network of shape[2,1062675]. After getting graphical data transformed, 96% of the data is used for train and 2% of positive and negative edges for test and validation set each. Thus, the obtained size of training samples after split is [2,509277] and validation and test samples of size [2, 10609] each.

4 ACKNOWLEDGEMENT

This article was written during the Data Science Lab 2020 at the University of Passau. Our team comprises:

Table 2: DSL2020 team members

Name	Primary Ownership of Phase
Akshat Sharma	Phase I Introduction
Lovesh Bishnoi	Phase II Data Acquisition and data Prepossessing
Mihir Shah	Phase III Model Implementation
Amit Manbansh	Phase IV Evaluation

[8]https://tkipf.github.io/graph-convolutional-networks/

REFERENCES

[1] Daniel A. Schult Aric A. Hagberg and Pieter J. Swart. Aug 2008. "Exploring network structure, dynamics, and function using NetworkX". *in Proceedings of the 7th Python in Science Conference (SciPy2008)* 1 (Aug 2008), pp. 11–15. http://conference.scipy.org/proceedings/SciPy2008/paper_2/

[2] Stryer L. Berg JM, Tymoczko JL. 2002. *Biochemistry*. Section 3.2: Primary Structure: Amino Acids Are Linked by Peptide Bonds to Form Polypeptide Chains, Vol. 5th edition. W H Freeman, New York. https://www.ncbi.nlm.nih.gov/books/NBK22364/

[3] Norberto de Souza O et al Breda A, Valadares NF. 2006 May 1 [Updated 2007 Sep 14]. Protein Structure, Modelling and Applications. *Bioinformatics in Tropical Disease Research: A Practical and Case-Study Approach [Internet].* 1 (2006 May 1 [Updated 2007 Sep 14]), Chapter A06. https://www.ncbi.nlm.nih.gov/books/NBK6824/

[4] Ul Qamar MT Chen LL Ding YD. Chang JW, Zhou YQ. Nov 22, 2016. Prediction of Protein-Protein Interactions by Evidence Combining Methods. *Int J Mol Sci. 2016;17(11):1946.* 6 (Nov 22, 2016). https://doi.org/10.3390/ijms17111946

[5] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[6] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein interface prediction using graph convolutional networks. In *Advances in neural information processing systems*. 6530–6539.

[7] S Jones and J M Thornton. 1996. Principles of protein-protein interactions. *Proceedings of the National Academy of Sciences* 93, 5 (1996), 13–20. https://doi.org/10.1073/pnas.93.1.13 arXiv:https://www.pnas.org/content/93/1/13.full.pdf

[8] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:cs.LG/1609.02907

[9] Xue Li, Lifeng Yang, Xiaopan Zhang, and Xiong Jiao. 2019. Prediction of Protein-Protein Interactions Based on Domain. *Computational and mathematical methods in medicine* 2019 (2019).

[10] Guo J Zhang DY Lin K. Wu X, Zhu L. 2006 Apr 26. "Prediction of yeast protein-protein interaction network: insights from the Gene Ontology and annotations.". *Nucleic Acids Resource* vol. 34,7 2137-50., 1 (2006 Apr 26). https://doi.org/10.1093/nar/gkl219

List of Figures

List of Tables