Predicting of Protein Interactions using Graph Convolutional Networks

Anonymous Author(s)

ABSTRACT

Predicting if a possible link exists between proteins is a standard task in network biology. This paper implements a link prediction technique in graph convolution networks (GCNs).

KEYWORDS

protein-protein interactions, link prediction, graph convolution networks

1 INTRODUCTION

Interactions between proteins are essential for all cellular processes, such as mediating metabolic pathways, signal transduction, gene expression and other cellular and systemic processes[24],[33]. A large amount of diseases are caused by protein mutations in regions responsible for interaction with other proteins which can lead to disruption of protein-DNA interaction, changes in protein folding patterns, unknown and undesirable interactions or enabling protein-pathogen interactions[11]. Ineractome mapping, i.e. mapping of complete protein-protein interactions (PPIs) within the organism is essential for understanding complex molecular relationships within living systems, as well as to elucidate various pathological conditions of the human organism.

In order to understand the mechanisms of interactions between proteins at the molecular level and to map PPIs at the global level, a large number of experimental techniques for the detection and characterization of protein interactions have been developed. Protein interactions can be identified using a variety of biochemical, genetic, and physical methods[28]. The most general classification of methods is based on the volume of data obtained - Small Scale Methods (SSM) and Large Scale Methods (LSM).

Small scale methods such as NMR and X-Ray crystallography[20] are used to (i) detect and identify protein interaction, (ii) intracellular visualization and (iii) verification, whereas large scale methods such as Yeast two-hybrid[14],[27], affinity purification/mass spectrometric identification methods such as TAP-tagging[13] and protein chips[31],[35] allow for a fast analysis of a large number of PPIs in a time efficient manner. However, these are expensive, sophisticated and time consuming methods, yielding poor accuracy of generated data[21].

Additionally, various machine learning methods have been extensively applied in protein-protein interaction prediction. The following methods have been shown as the most successful and some of them are: support vector machine (SVM)[19][3], neural networks (NN)[34][9] and Bayesian networks (BN)[25][2].

Recent trend in representing the data in the form of graphs has made it easier for analysis and various prediction tasks. Graph convolutional networks (GCN), which are based on classic convolutional networks for graph structured data are especially gaining attention due to their applicability to various problems such as node classification[6], link prediction[8][29], community detection[5] and in social recommendation systems[6].

Problem Statement. In the context of protein-protein interactions, predicting if a possible link between nodes exist involves very expensive, and time consuming experiments. Available methods shows many different algorithms working well in practice, however little attention is paid to their limitations when interaction network data is incomplete or noisy[7].

Objective. We aim to understand the applicability of graph convolutional networks in the link prediction problem in the context of protein-protein interactions.

Contribution. We extend the work presented in [18] as the link prediction problem on undirected and unweighted networks. In an experimental setup we analyse the performance of the two-layer GCN model on a real-world dataset. Furthermore, we investigate and compare different hyperparameter settings in the context of model accuracy.

Outline. The remainder of this paper is organized as follows: Section 2 provides a short review on state of the art in PPIs. In Section 3 we formally define the problem and give a brief introduction to graph convolutional networks, followed by research design and questions in 1.3. Sections 2 and 3 present data preprocessing and model implementation procedure before we evaluate and conclude the paper in 4.

2 RELATED WORK

According to the work done in [30], PPI prediction approaches are classified as follows:

(1) Physiocemical experimental and
(2) Computational approaches.

In Section 1 we briefly mentioned several experimental techniques for PPI prediction such as yeast two-hybrid[14],[27] and protein chips[31],[35]. These techniques fall into physiochemical experimental techniques and thus they identify physiochemical interactions between proteins which in turn are used to predict the relationships between them.

Computational approaches can be classified into sequence-based and structure-based approaches based on their protein features[30].

In this Section we focus on the computational approaches and describe some of the methods in detail.

2.1 Sequence-Based Approaches

Mirror Tree Method is a sequence-based statistical method proposed by Pazos and Valencia in [26]. The method compares evolutionary distances between sequences of the associated proteins and uses topological similarity of phylogenetic trees to predict PPIs. The distance is taken as the average value of the residue similarities from the McLachlan amino acid homology matrix[23] and the similarity between trees is simply the correlation between the distance matrices. This approach is independent from any other tree-construction methods and only involves analyzing the distance matrices and no tree-creation is necessary for PPI prediction.

Gao et al. in [12] introduced another sequence-based method that uses Machine Learning (ML) in PPI prediction. The method combines auto covariance feature representation and Support Vector Machine (SVM) and has been evaluated on the yeast S. *cerevisiae* dataset achieving very promising prediction results[12] with average prediction accuracy, sensitivity, and precision of 0.86, 0.85, and 0.87 respectively.

2.2 Structure-Based Approaches

In [4] authors Chen and Liu introduce a structure-based ML approach using random forest of decision trees predict PPIs. The method is capable of exploring all possible domain interactions and making predictions based on all the protein domains. Experimental results show that they can predict PPIs with higher sensitivity (79.78%) and specificity (64.38%).

3 BACKGROUND

3.1 Problem Statement

As stated in [22] an undirected network can be represented as $\mathcal{G}(\mathcal{V},\mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Multiple links and self-connections are not allowed. The universal set $\mathcal{U}$ containing all possible links is represented as

$$\frac{|\mathcal{V}| \cdot (|\mathcal{V}| - 1)}{2} \tag{1}$$

where $|\mathcal{V}|$ denotes the number of elements in the set $\mathcal{V}$. Accordingly, the set of non-existing links is represented as $\mathcal{U} - \mathcal{E}$. Under the assumption that there are missing links in the set $\mathcal{U} - \mathcal{E}$ the task of link prediction is to detect them.

For testing the algorithm's accuracy, the set of observed links, $\mathcal{E}$, is randomly divided into two parts: the training set, $\mathcal{E}^{\mathcal{T}}$ and is treated as known information, while the validation set, $\mathcal{E}^{\mathcal{P}}$ is used only for testing.

The advantage of this sub-sampling is the independence of the training split on the number of iterations, but poses a threat to statistical conclusion validity since some links may never be selected in the probe set, whereas some may be included more than once. However, these limitations can be overcame using K-*fold cross-validation.* In this sampling method, the observed links are randomly partitioned into K subsets. Each time one subset is selected as probe set, the rest $K-1$ constitute the training set. The process is then repeated K times, with each of the K subsets used exactly once as the probe set. This method in contrast to random-sampling method ensures that all the links are used in both testing and validation and links for the prediction are used only once.

The standard metrics for evaluating the accuracy of the prediction algorithms: *area under the receiver operating characteristic curve* (AUC) and *average precision* (AP) are discussed in Section 4.

3.2 Graph Convolutional Networks

In [18] the goal for GCNs is to learn a function of features on a graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$ with following inputs:

- A feature description x_i for every node i which is represented in a N × D feature matrix where N denotes a number of nodes and D, the number of input features
- adjacency matrix A as a representative description of the graph structure

to produce a an output Z, an $N \times F$ feature matrix. The definition of neural network layers can be represented as a non-linear function

$$H^{(l+1)} = f(H^{(l)}, A) \tag{2}$$

$$H^{(0)} = X \tag{3}$$

$$H^{(L)} = Z \tag{4}$$

with L being the number of layers used.

GCNs rely on (i) Attention Mechanisms, to maintain state information to capture the neighbourhood properties of nodes which makes to remember important nodes and gives them higher weights throughout the learning process (ii) Graph spatial-Temporal Network can support graphs with a fixed structure but inputs with that change over time.

Figure 1 shows a simple representation of the GCN layers with first-order filters.

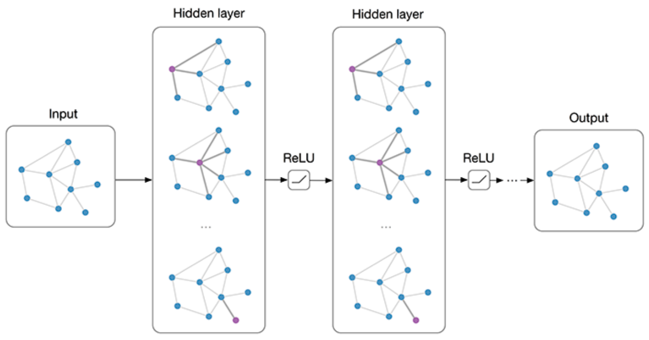

Figure 1: Figure taken from Kipf[16] and shows multi-layer GCN with first-order filters.

3.3 Research Design

We present the research design in Figure 2 with an overview of the overall research method, which will be explained in detail in subsequent sections.

3.4 Research Objective and Questions

Our overall objective is to investigate the applicability of GCNs in link prediction problem. We implement the graph convolutional network as a method in link prediction problem of protein-protein interactions. Additionally, we compare the model accuracy under the different hyperparameter settings. For this, we pose the research following research questions presented in Table 1.

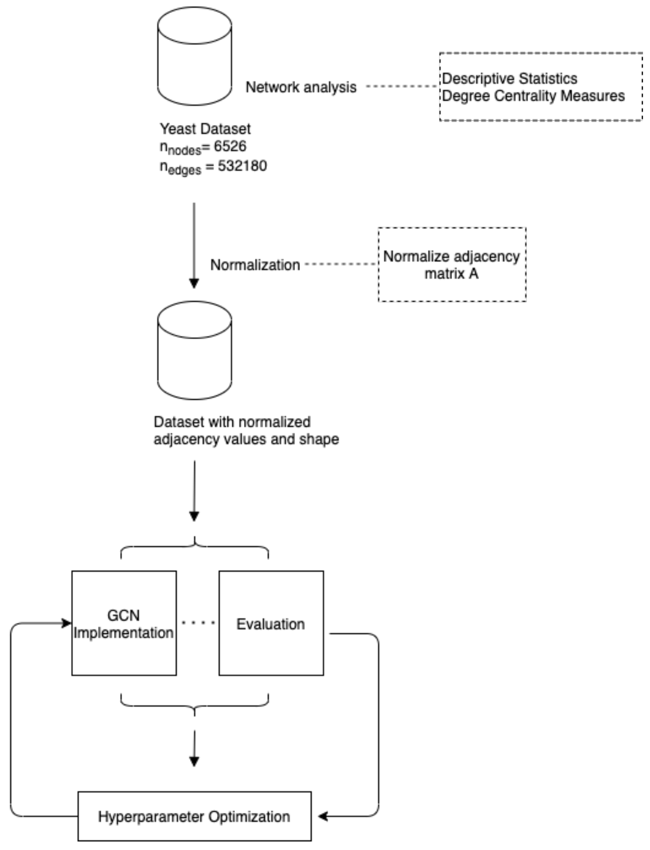
Network analysis
Descriptive Statistics
Degree Centrality Measures
Yeast Dataset
n_nodes= 6526
n_edges = 532180
Normalization
Normalize adjacency matrix A
Dataset with normalized adjacency values and shape
GCN Implementation
Evaluation
Hyperparameter Optimization

Figure 2: Overview of the research method used in this work.

Table 1: Overview of the research questions of this study

	Research Question
RQ1	*Which hyperparameters in GCNs matter the most for the model performance?* In the first step we study and investigate different parameters (epochs, learning rate and hidden units) and their effect on the overall model performance.
RQ2	*Which values of these hyperparameters are optimal for high performance?* In the second step we tune the values of the parameters in RQ1 and analyze their effect on the overall model performance.

4 DATA ACQUISITION & PRE-PROCESSING

4.1 Data acquisition

In this paper, we consider the yeast protein-protein interaction dataset[1] which is already preprocessed and ready for analysis. This dataset was collected for research purposes at the Stanford Network Analysis Project[2] (SNAP) and is available at SNAP's Biomedical Dataset Collection[3].

[1]http://snap.stanford.edu/deepnetbio-ismb/ipynb/yeast.edgelist
[2]https://snap.stanford.edu/about.html
[3]https://snap.stanford.edu/biodata/index.html

4.2 Frameworks & Libraries

We use Python v. 3.6[4] as the standard language of implementation due to it's vast amount of libraries available. We give the overview of all other packages used including the version at the time of writing in Table 2.

Table 2: Libraries

Networkx	v2.4
Numpy	v1.18.5
torch	1.5.1+cu101
Matplotlib	v3.3.2
seaborn	v0.11.0

4.3 Network Analysis

We use yeast PPI network dataset from SNAP. The network consists of different proteins and their interactions that have been previously verified through experimental techniques. We summarize main and most relevant network information in Table 3.

Table 3: Graph Info

Data Source	'yeast.edgelist'[5]
Node type	Protein
Edge type	Interaction
Format	⋃ Undirected
Edge weights	− Unweighted
Size	6526 nodes (proteins)
Volume	532180 edges (interactions)
Average degree	163.0953

4.3.1 *Network Density.*

Network density represents the fraction of edges present over all possible edges in a network. For undirected graphs, density can be calculated using the equation bellow.

$$\eta = \frac{2|E|}{|V|(|V|-1)} \tag{5}$$

with $|V|$ being the number of vertexes and $|E|$ number of edges. A graph is complete if there exists an edge between every pair of vertexes and it's density is therefore 1. For our graph the obtained network density is **0.024** indicating that it is not a very dense graph.

4.4 Centrality Measures

4.4.1 *Degree Centrality.*

In network science, the degree centrality is one of the many metrics that can be used to evaluate the importance of a node. The degree of a node is a number of nodes that a given node is connected to[10] and can be calculated as follows:

$$k_i = \sum_{j=1}^{n} A_{ij} \tag{6}$$

[4]https://www.python.org/downloads/release/python-360/

where A is the adjacency matrix and i, j are nodes. For example, if there exists an edge between nodes i and j then $A_{ij} = 1$ and 0 otherwise.

Figure 3 shows the degree centrality distribution for our dataset.

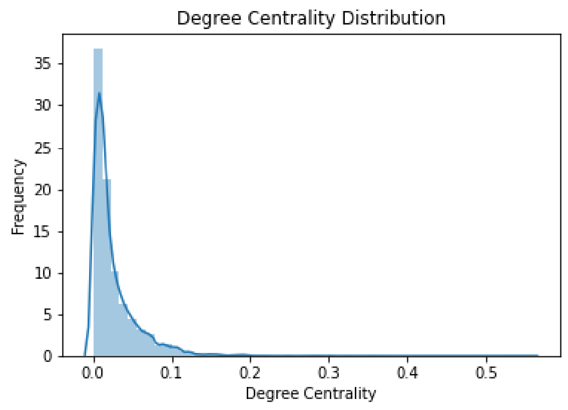

Figure 3: Degree Centrality Distribution of the Yeast dataset

4.4.2 Betweenness Centrality. In [10] the betweenness centrality of a node is the number of shortest paths between other nodes that run through the node in interest and it can be expressed as follows:

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{7}$$

where σ_{st} is the total number of shortest paths from node s to node t and $\sigma_{st}(v)$ is the number of paths that pass through v.

Figure 4 shows the betweenness centrality for our dataset and Figure 5 represents both degree and betweenness distributions (for comparison).

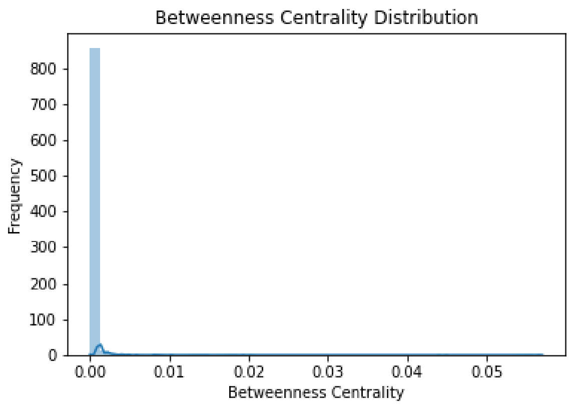

Figure 4: Betweenness Centrality Distribution of the Yeast dataset

We also present five most important nodes (proteins) computed with these centrality measures in Table 4 and for example the left side of the table shows proteins which have the most interactions in our network. However for a more meaningful interpretation we lack experience and expertise in the domain for network biology.

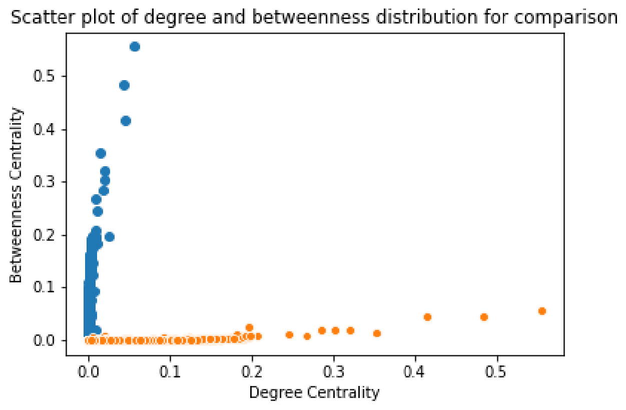

Figure 5: Betweenness and Degree Centrality Distributions of the Yeast dataset

Table 4: Top 5 proteins computed with Degree and Betweenness Centrality Measures

Protein	Degree	Protein	Betweenness
YDL160C	3619	YDL160C	0.05
YAL021C	3155	YGL122C	0.04
YGL122C	2712	YAL021C	0.04
YFL039C	2300	YPR042C	0.02
YNL209W	2085	YNL209W	0.02

5 MODEL IMPLEMENTATION

5.1 Approach

We previously defined our network as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $N = |\mathcal{V}|$ nodes. The N represents the number of proteins and each vertex of $\mathcal{G}$ represents a single protein while each edge represents the interaction. Furthermore, given an adjacency matrix A of $\mathcal{G}$ is given and A_{ij} denotes if there exists an interaction between proteins i and j.

With the notation above, our model follows simple encoder-decoder approach. The role of the encoder is to map each node v_i to a low-dimensional vector embedding z_i. The decoder then extracts the information necessary like classification label associated between two proteins. This approach was introduced by Kipf in [17] and allows for decoding high-dimensional graph information from encoded low-dimensional embeddings. The representation architecture for this approach is shown in Figure 6.

We now introduce relevant parts of our model implementation in detail.

5.2 Encoder

Encoder is essentially a function (see equation 8):

$$\mathrm{ENC} : V \longrightarrow \mathbb{R}^{\mathrm{d}} \tag{8}$$

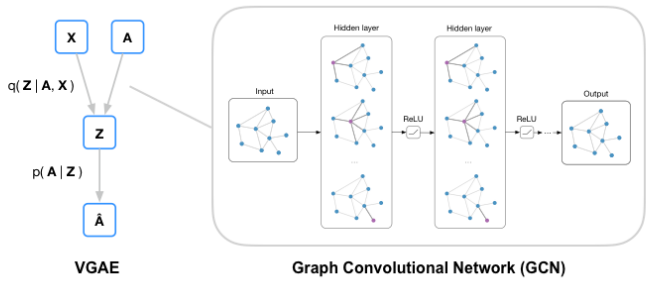

Figure 6: Graph Auto-Encoders Model Architecture taken from [18]

with a goal of mapping nodes to vector embeddings, $\mathbf{z}_i \in \mathrm{R}^{\mathrm{d}}$, where z_i represents the embedding for a node v_i.

We closely follow the work from Kipf in [17] and define our two-layer GCN model as follows:

$$q(\mathrm{Z} \mid \mathrm{X}, \mathrm{A}) = \prod_{i=1}^{N} q\left(\mathbf{z}_i \mid \mathrm{X}, \mathrm{A}\right) \tag{9}$$

,

$$q\left(\mathbf{z}_i \mid \mathrm{X}, \mathrm{A}\right) = \mathcal{N}\left(\mathbf{z}_i \mid \boldsymbol{\mu}_i, \operatorname{diag}\left(\sigma_i^2\right)\right) \tag{10}$$

In Equation 9 and 10 $\boldsymbol{\mu} = \mathrm{GCN}_{\mu}(\mathbf{X}, \mathbf{A})$ is the matrix of mean vectors $\boldsymbol{\mu}_i$ and $\log \boldsymbol{\sigma} = \mathrm{GCN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A})$.

Similarly, the two-layers GCN model is defined as:

$$\mathrm{GCN}(\mathbf{X}, \mathbf{A}) = \tilde{\mathbf{A}}\,\mathrm{ReLU}\left(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_0\right)\mathbf{W}_1 \tag{11}$$

where $\boldsymbol{W}_{\boldsymbol{i}}$ are parameter matrices that need to be trained. The idea behind ReLu activation function in Equation 11 is that for every node and it's neighbouring node, the feature vector is summed up and in this way GCN can learn embeddings effectively through integrating neighbouring graph features.

5.3 Pairwise Decoder

In [17] acts as a generative model and it's a simple inner product decoder defined as:

$$p(\mathbf{A} \mid \mathbf{Z}) = \prod_{i=1}^{N} \prod_{j=1}^{N} p\left(A_{ij} \mid \mathbf{z}_i, \mathbf{z}_j\right) \tag{12}$$

with

$$p\left(A_{ij} = 1 \mid \mathbf{z}_i, \mathbf{z}_j\right) = \sigma\left(\mathbf{z}_i^{\top} \mathbf{z}_j\right) \tag{13}$$

, where A_{ij} are the elements of adjacency matrix A and $\sigma(\cdot)$ is the logistic sigmoid function.

5.4 Training details

We use weighted cross entropy with logits[6] as our loss function. Additionally we add Kullback-Leibler divergence to our loss function as introduced in [15] in order to minimize the the reconstruction error of adjacency matrix A. The loss function can formally be represented as follows:

$$= \mathbb{E}_{q(Z|X,A)} \log p\left(A^{*} \mid Z\right)\big] - KL[q(Z \mid X, A) \| p(Z)] \tag{14}$$

.

[6]https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/weighted_cross_entropy_with_logits

5.5 Experimental results on link prediction

We implemented our model using PyTorch in Python and used free resources from Google Colab[7] to train our model. The experimental model has two hidden layers with 32 and 16 neurons respectively. We trained initially for 20 iterations using Adam algorithm with a learning rate of 0.01.

The results for all runs in this phase and individual model settings can be found in the *Outputs* directory at our GitLab repository[8].

6 EVALUATION

In this section, we present in detail the metrics that are used to evaluate a link prediction task. Furthermore, we conduct a small hyperparameter study and tune a different number of parameters.

6.1 Link Prediction Performance

6.1.1 Area under the ROC Curve (AUC). AUC measures the whole two-dimensional area underneath the ROC curve, see Figure **??**. Receiver operating characteristic or ROC curve defines a curve of true positive rate or recall (see equation 15) vs. false positive rate (see equation 16) at different classification thresholds.

$$\textit{True Positive Rate} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}} \tag{15}$$

$$\textit{False Positive Rate} = \frac{\text{False Positives}}{\text{False Positives + True Negatives}} \tag{16}$$

AUC provides a measure of performance across all possible classification thresholds. We can interpret AUC score as the likelihood of the model ranking a random positive example more highly than a random negative example. AUC scores range from 0 to 1, therefore, a model with all correct predictions has a score of 1 and a model whose predictions are wrong has a score of 0.

6.1.2 Average Precision (AP). AP is defined as a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight:

$$\mathrm{AP} = \sum_{n} \left(R_n - R_{n-1}\right) P_n \tag{17}$$

where P_n and R_n are the precision and recall at the nth threshold. AP scores range from 0 to 1, and a higher score is better.

6.2 Hyperparameter Optimization

In machine learning hyperparameters play an important role since they directly control the behaviours of training algorithms and significantly affect the performance of machine learning models. Paper [32] mentions 2 hyperparameter optimization methods: manual search and automatic search methods.

Manual search includes selecting hyperparameters intuitively, by hand. This approach is very limited as it requires a significant amount of expertise and professional background knowledge for identifying important parameters that somehow influence the results or the performance of the machine learning mode.

[7]https://colab.research.google.com/
[8]https://git.fim.uni-passau.de/padas/20ss-data-science-lab/team16/predicting-of-protein-interactions-using-gcns

Table 5: Proposed GCN Search Space

Units in the first hidden layer	256, 128, 64, 32
Units in the second hidden layer	64, 32, 16, 8
Epochs	5, 10, 15, 20
Learning rate	0.001, 0.01, 0.1

Automatic search methods, such as grid search[1] have been proposed with a goal of overcoming the drawbacks in manual search. The idea behind grid search is so-called *exhaustive search*. Grid search is a brute force method which trains a machine learning model for all possible values in the predefined hyperparamer search space. This method is able to achieve optimal scores with automatic tuning, however it's efficiency decreases rapidly as the search space, or range of possible values of the hyperparameters increases.

6.3 GCN Hyperparameters

Tuning neural networks can be considered a difficult task especially when the hyperparameters are chosen poorly. This leads to a network to learn really slow or in worst case to not learn at all. (IDK)

For the problem of link prediction in GCN we consider tuning of the following parameters: *number of epochs, number of units in hidden layers and learning rate*. We briefly explain the importance of the following parameters bellow.

(1) NUMBER OF EPOCHS: An epoch is defined as a full pass of the dataset. Setting an epoch to a small number may not give a neural network enough time to learn good parameters. Similarly, setting an epoch to a really big number may overfit the training data.
(2) LEARNING RATE: Learning rate is considered to be one of the most parameters in machine learning. Again setting the learning rate as too small or too large may cause a problem in learning phase of the neural network. Consequently, it can lead to really slow learning or not learning at all.
(3) UNITS IN HIDDEN LAYERS: Deciding a number of neurons in hidden layers plays an important role in network architecture. Having too few neurons results in underfitting, meaning that there is not a sufficient number of neurons to adequately detect the signals in the dataset. Similarly, having too many neurons results in overfitting.

6.3.1 Proposed Hyperparameter Search Space. The values of the hyperparameters used in tuning are presented in Table 5.

7 ACKNOWLEDGEMENT

This report was written during the Data Science Lab 2020 at the University of Passau. Our team members and individual responsibilities can be found in Table 6.

Table 6: DSL2020 team members

Phase	Name
I)	Aleena Elsa George
II)	Lavannya Varghese
III)	Dzejlana Karajic
IV)	Samyuktha Lakshmi Maddela

REFERENCES

[1] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* 13, null (Feb. 2012), 281–305.

[2] James R. Bradford, Chris J. Needham, Andrew J. Bulpitt, and David R. Westhead. 2006. Insights into Protein–Protein Interfaces using a Bayesian Network Prediction Method. *Journal of Molecular Biology* 362, 2 (2006), 365 – 386. https://doi.org/10.1016/j.jmb.2006.07.028

[3] James R. Bradford and David R. Westhead. 2004. Improved prediction of protein–protein binding sites using a support vector machines approach. *Bioinformatics* 21, 8 (12 2004), 1487–1494. https://doi.org/10.1093/bioinformatics/bti242 arXiv:https://academic.oup.com/bioinformatics/article-pdf/21/8/1487/697068/bti242.pdf

[4] Xue-Wen Chen and Mei Liu. 2005. Prediction of protein–protein interactions using random decision forest framework. *Bioinformatics* 21, 24 (10 2005), 4394–4400. https://doi.org/10.1093/bioinformatics/bti721 arXiv:https://academic.oup.com/bioinformatics/article-pdf/21/24/4394/433774/bti721.pdf

[5] Zhengdao Chen, Lisha Li, and Joan Bruna. 2019. Supervised Community Detection with Line Graph Neural Networks. *arXiv: Machine Learning* (2019).

[6] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. 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 (KDD '19)*. Association for Computing Machinery, New York, NY, USA, 257–266. https://doi.org/10.1145/3292500.3330925

[7] Berend Snel Michael Cornell Stephen G. Oliver Stanley Fields Peer Bork Christian von Mering, Roland Krause. 2002. A comprehensive analysis of protein–protein interactions in Saccharomyces cerevisiae. *Nature* 417 (2002), 399–403. https://doi.org/10.1038/nature750

[8] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2D Knowledge Graph Embeddings. *CoRR* abs/1707.01476 (2017). arXiv:1707.01476 http://arxiv.org/abs/1707.01476

[9] Piero Fariselli, Florencio Pazos, Alfonso Valencia, and Rita Casadio. 2002. Prediction of protein–protein interaction sites in heterocomplexes with neural networks. *European Journal of Biochemistry* 269, 5 (2002), 1356–1361. https://doi.org/10.1046/j.1432-1033.2002.02767.x arXiv:https://febs.onlinelibrary.wiley.com/doi/pdf/10.1046/j.1432-1033.2002.02767.x

[10] Linton C Freeman. 1978. Centrality in social networks conceptual clarification. *Social networks* 1, 3 (1978), 215–239.

[11] Mileidy W. Gonzalez and Maricel G. Kann. 2012. Chapter 4: Protein Interactions and Disease. *PLOS Computational Biology* 8, 12 (12 2012), 1–11. https://doi.org/10.1371/journal.pcbi.1002819

[12] Yanzhi Guo, Lezheng Yu, Zhining Wen, and Menglong Li. 2008. Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences. *Nucleic Acids Research* 36, 9 (04 2008), 3025–3030. https://doi.org/10.1093/nar/gkn159 arXiv:https://academic.oup.com/nar/article-pdf/36/9/3025/16752779/gkn159.pdf

[13] Gruhler A. Heilbut A. et al. Ho, Y. 2002. Systematic identification of protein complexes in Saccharomyces cerevisiae by mass spectrometry. *Nature* 415 (2002), 180–183. https://doi.org/10.1038/415180a

[14] Takashi Ito, Tomoko Chiba, Ritsuko Ozawa, Mikio Yoshida, Masahira Hattori, and Yoshiyuki Sakaki. 2001. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences* 98, 8 (2001), 4569–4574. https://doi.org/10.1073/pnas.061034498 arXiv:https://www.pnas.org/content/98/8/4569.full.pdf

[15] Diederik P. Kingma and M. Welling. 2014. Auto-Encoding Variational Bayes. *CoRR* abs/1312.6114 (2014).

[16] Thomas Kipf. 2016. *Graph Convolutional Networks*. Retrieved September 20, 2020 from https://tkipf.github.io/graph-convolutional-networks/

[17] Thomas Kipf and M. Welling. 2016. Variational Graph Auto-Encoders. *ArXiv* abs/1611.07308 (2016).

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

[19] Asako Koike and Toshihisa Takagi. 2004. Prediction of protein–protein interaction sites using support vector machines. *Protein Engineering, Design and Selection* 17, 2 (02 2004), 165–173. https://doi.org/10.1093/protein/gzh020 arXiv:https://academic.oup.com/peds/article-pdf/17/2/165/4509864/gzh020.pdf

[20] Sylvie Lalonde, David W. Ehrhardt, Dominique Loqué, Jin Chen, Seung Y. Rhee, and Wolf B. Frommer. 2008. Molecular and cellular approaches for the detection of protein–protein interactions: latest techniques and current limitations. *The Plant Journal* 53, 4 (2008), 610–635. https://doi.org/10.1111/j.1365-313X.2007.03332.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1365-313X.2007.03332.x

[21] Zheng-Wei Li, Zhu-Hong You, Xing Chen, Li-Ping Li, De-Shuang Huang, Gui-Ying Yan, Ru Nie, and Yu-An Huang. 2017. Accurate prediction of protein-protein interactions by integrating potential evolutionary information embedded in PSSM profile and discriminative vector machine classifier. *Oncotarget* 8, 14 (2017), 23638–23649. https://doi.org/10.18632/oncotarget.15564

[22] Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications* 390, 6 (2011), 1150 – 1170. https://doi.org/10.1016/j.physa.2010.11.027

[23] A.D. McLachlan. 1971. Tests for comparing related amino-acid sequences. Cytochrome c and cytochrome c551. *Journal of Molecular Biology* 61, 2 (1971), 409 – 424. https://doi.org/10.1016/0022-2836(71)90390-1

[24] Yi Lu Manman Zhhao Mengying Zhang, Qiang Su and Bing Niu. 2017. Application of Machine Learning Approaches for Protein-protein Interactions Prediction. *Medicinal Chemistry* 13, 6 (2017), 506–514. https://doi.org/10.2174/1573406413666170522150940

[25] Hani Neuvirth, Ran Raz, and Gideon Schreiber. 2004. ProMate: A Structure Based Prediction Program to Identify the Location of Protein–Protein Binding Sites. *Journal of Molecular Biology* 338, 1 (2004), 181 – 199. https://doi.org/10.1016/j.jmb.2004.02.040

[26] Florencio Pazos and Alfonso Valencia. 2001. Pazos, F. Valencia, A. Similarity of phylogenetic trees as indicator of protein-protein interaction. Protein Eng. 14, 609-614. *Protein engineering* 14 (10 2001). https://doi.org/10.1093/protein/14.9.609

[27] Gerard Cagney Traci A. Mansfield Richard S. Judson James R. Knight Daniel Lockshon Vaibhav Narayan Maithreyan Srinivasan Pascale Pochart Alia Qureshi-Emili Ying Li Brian Godwin Diana Conover Theodore Kalbfleisch Govindan Vijayadamodar Meijia Yang Mark Johnston Stanley Fields Jonathan M. Rothberg Peter Uetz, Loic Giot. 2000. A comprehensive analysis of protein–protein interactions in Saccharomyces cerevisiae. *Nature* 403 (2000), 623–627. https://doi.org/10.1038/35001009

[28] Sujini GN Kumar GN. Rao VS, Srinivas K. 2014. Protein-protein interaction detection: methods and analysis. *International journal of proteomics* (2014). https://doi.org/10.1155/2014/147648

[29] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *The Semantic Web*, Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam (Eds.). Springer International Publishing, Cham, 593–607.

[30] Maad Shatnawi. 2015. Review of Recent Protein-Protein Interaction Techniques. (12 2015), 99–121. https://doi.org/10.1016/B978-0-12-802508-6.00006-5

[31] Amy Hin Yan Tong, Becky Drees, Giuliano Nardelli, Gary D. Bader, Barbara Brannetti, Luisa Castagnoli, Marie Evangelista, Silvia Ferracuti, Bryce Nelson, Serena Paoluzi, Michele Quondam, Adriana Zucconi, Christopher W. V. Hogue, Stanley Fields, Charles Boone, and Gianni Cesareni. 2002. A Combined Experimental and Computational Strategy to Define Protein Interaction Networks for Peptide Recognition Modules. *Science* 295, 5553 (2002), 321–324. https://doi.org/10.1126/science.1064987 arXiv:https://science.sciencemag.org/content/295/5553/321.full.pdf

[32] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. 2019. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimizationb. *Journal of Electronic Science and Technology* 17, 1 (2019), 26 – 40. https://doi.org/10.11989/JEST.1674-862X.80904120

[33] Jian Zhang and Lukasz Kurgan. 2017. Review and comparative assessment of sequence-based predictors of protein-binding residues. *Briefings in Bioinformatics* 19, 5 (03 2017), 821–837. https://doi.org/10.1093/bib/bbx022

[34] Huan-Xiang Zhou and Yibing Shan. 2001. Prediction of protein interaction sites from sequence profile and residue neighbor list. *Proteins: Structure, Function, and Bioinformatics* 44, 3 (2001), 336–343. https://doi.org/10.1002/prot.1099 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/prot.1099

[35] Heng Zhu, Metin Bilgin, Rhonda Bangham, David Hall, Antonio Casamayor, Paul Bertone, Ning Lan, Ronald Jansen, Scott Bidlingmaier, Thomas Houfek, Tom Mitchell, Perry Miller, Ralph A. Dean, Mark Gerstein, and Michael Snyder. 2001. Global Analysis of Protein Activities Using Proteome Chips. *Science* 293, 5537 (2001), 2101–2105. https://doi.org/10.1126/science.1062191 arXiv:https://science.sciencemag.org/content/293/5537/2101.full.pdf