SUMMARY ARTICLE

Applying Graphical Neural Networks to PPI Dataset

Isaac Akintaro*

London, UK
*Corresponding author. Email: isaacakintaro@gmail.com

1. Context

The Protein-Protein Interaction (PPI) dataset describes the physical and functional interactions between different proteins in a biological system. The significance of this problem is that PPI provides valuable information for the development of drugs for various diseases. For example, identifying proteins that interact with disease-causing proteins can help in the design of drugs that can block these interactions and thus cure the disease.

The dataset given to me consists of 24 tissues described by 51,420 protein instances. Each protein is described by 50 features and these map to 121 tissue-specific ontology cellular functions that the protein exibits in that particular tissue. The tissues are split into training (20 tissues), cross validation (2 tissues) and testing (2 tissues).

Proteins exhibit different cellular functions in different tissues, so it is very challenging to predict their behaviour. However, proteins interact with each other, capturing these relationships in neighbourhoods would provides a better understanding of how they would behave in a different tissue. Hence the need for Graphical Neural Networks to model the interactions of the different proteins. Each tissue would be a graph, the proteins are the nodes and the edges would be the connections between the proteins.

Figure 1 shows the neighbourhood of a specific protein across the entire training set. This visualisation shows that self-loops exist and will need to be filtered in the data processing.

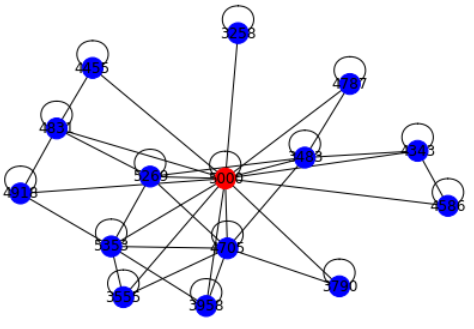

Figure 1. Visualisation of a neighbourhood of node id 5000 in the training set.

2. Method

2.1 Libraries and Frameworks

I experimented with Jupyter Labs trying both Tensorflow GNN libraries and PyTorch Geometric, I had issues installing specific modules and hence opted for Google Colab which runs freely over the

cloud without much setup.

On trying to apply lessons from Tensorflow GNN tutorials I found it tricky to format the data into a Graph object that could be read and used. PyTorch Geometric had more support for this aswell as example code that I have used which made the whole process more efficient.

2.2 Data Preprocessing

I stored the raw data provided in a folder on my Google Drive. I was able to mount my Google Drive and access the data for processing converting into PyTorch tensor files (.pt files). Data objects were created for each tissue graph and stored, these contained the nodes, edges (with self-looping removed), features and cellular functions labels.

2.3 Evaluation Metric

My dataset of proteins has 121 cellular functions as possible labels. Each protein node could have 0, 1 or more of these cellular functions. Some cellular function classes may have very large instances compared to others i.e. imbalanced distribution of classes. To investigate this I analysed our training data and found at the extremes that we had two class that appeared almost ten times more than another class as shown in the top two rows of Table 1.

Table 1. Top 5 and Bottom 5 classes of cellular functions proteins could exhibit in our training dataset.

Top Five Class IDs	Count	Bottom Five Class IDs	Count
32	39932	86	4127
116	39828	16	4651
118	36647	65	5564
117	34288	4	5753
12	31890	77	5930

Using accuracy to train my model may not be wise as it might always predict the cellular function classes that have the most instance to get a high accuracy, for example Class ID 32 and 116 show up 90% of the time in the training dataset.

Therefore F1 Score which is made up of precision and recall is a more balanced metric which has been used for training. The F1 score is defined in Equation 1. The precision is defined in Equation 2. The recall is defined in Equation 3.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{1}$$

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{3}$$

2.4 Task Architecture Selection

Following examples in PyTorch Geometric I attempted multi-label node classification where the 121 tissue-specific cellular functions of each protein were predicted. For this I experimented on two deep learning architectures for graph-structured data.

1. `GAT-Conv`
   Graph Attention Convolutional Networks, where convolutions operations are performed using a graph attention mechanism to weigh the importance of each node in the graph.

2. `FiLM-Conv`
   Feature-wise Linear Modulation Convolutional Networks, where a feature-wise linear modulation layer is added to the traditional ConvNet architecture. This layer applies a separate linear transformation to input features which allows to capture relationships between the features, weighing each feature differently leading to more accurate predictions.

These two architectures were trained on the same data with a learning rate of 0.005 and a training batch size of 2 tissue graphs. Their performance is shown in Figures 2 and 3 below. The GAT GNN has a higher performance 0.96 vs FiLM that had a performance of 0.88 for the F1 Score. The GAT architecture shows a 9% improvement on the FiLM architecture and had a smoother training loss slope. In the next section I explore hyperparameter tuning on the GAT model.

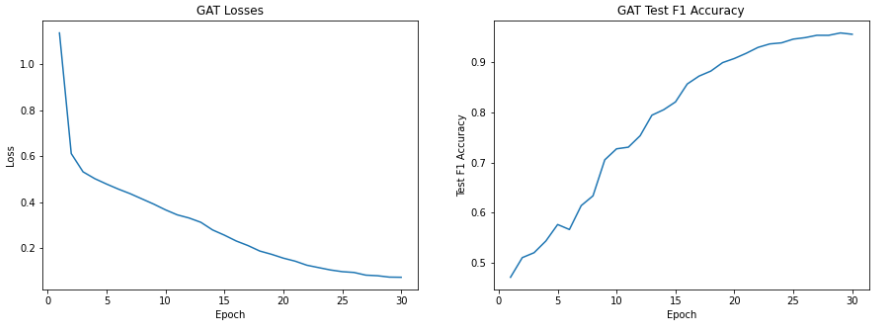

Figure 2. GAT Performance for Multi-Label Node Classification.

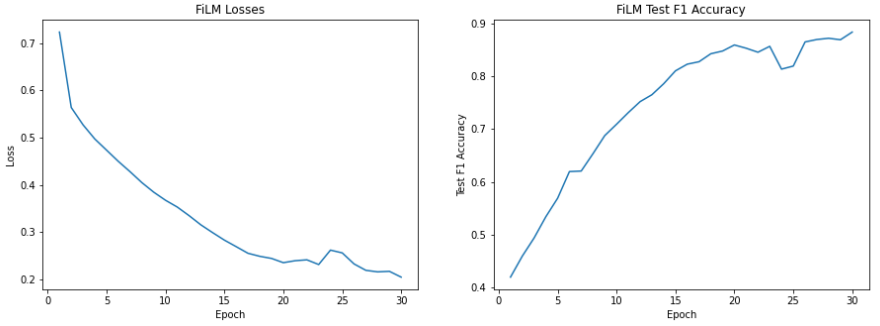

Figure 3. FiLM Performance for Multi-Label Node Classification.

According to the PapersWithCode 2023b Leaderboard GAT is able to reach an F1 performance of 0.97 for the node classification task on the PPI dataset. With more hyperparameter tuning this should be possible for the model.

With more time I would also like to try other architectures such as GCN2Conv, Cluster GCN and GeniePath GNNs but wasn't able to make them work under time constraints.

I also attempted Link Prediction using a neural network architecture called GraphSAGE which samples a set of neighbor nodes for each node and aggregates the features of these neighbors to obtain a representation of the node (unlike other architectures which don't engage sampling). In Link Prediction, the edge connection of a protein to another protein is evaluated. The performance is shown in Figure 4. It plateaus very quickly and the F1 Score oscillates around 0.52. According to PapersWithCode 2023b GraphSAGE had a F1 performance of 0.61 for node classification, so it is expected to perform worse for the task of link prediction (due to the incompleteness of the underlying graph). The best for link prediction according to PapersWithCode 2023a was an architecture called HGCN which had a score of 0.85.

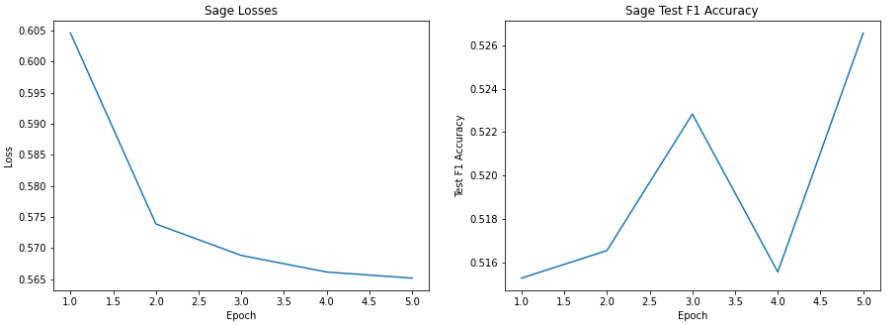

Figure 4. GraphSAGE Performance for Link Prediction.

2.5 What are the GNN layers doing?

In a Graphical Neural Network each layer processes the features of the nodes in the graph by aggregating this information and information from their neighbouring nodes and updating the node's representation in the new layer. Through this architecture, the network is able to take into account the relationships betweeen nodes and their neighbours. This powerful for helping us understand how the proteins will behave in different tissues which is influenced by their neighbouring proteins. The GNN layers preserve graph structure while transforming node features in a way that can allow predictions to be performed.

3. Evaluation

3.1 Observations and Findings

Hyperparameter tuning was performed by varying the learning rate on the GAT architecture as shown in Figure 5. A learning rate of 0.02 overfits the training data and fails to generalise for the test set and would need a longer training time through the dataset. The learning rate of 0.002 had the best F1 Accuracy for the test set of 0.97. 30 epochs also seems the right amount of passing through the data set before it plateaus.

This experiment took about 4.5 hours to run, about over an hour per learning rate. Without time constraints I would like to vary other hyperparameters such as activation functions, number of layers, number of units in our hidden layers and the training batch size. I could use the GridSearchCV class in scikit-learn to conduct these hyperparameter tunings. With more time I would also like to understand what features have been prioritised in our GAT model.

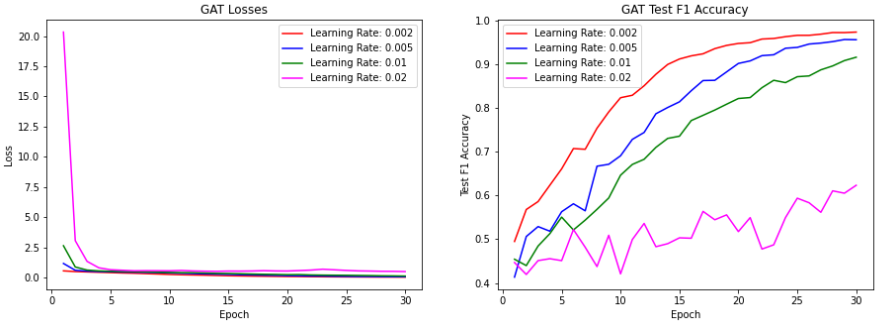

Figure 5. GAT Learning Rate Experimentation for Multi-Label Node Classification.

3.2 Assumptions

A key Machine Learning assumption used to describe data, is that the data samples is independent of other data samples and each sample is drawn from the same underlying distribution. This might not be the case as the paper Zitnik and Leskovec 2017 refers to 107 human tissues which are modelled as graphs, we used only 24, these tissues/graphs could all be to do with one organ and may be overfitted and not generalise well to other tissues in other organs. Also in the paper one of the key assumptions made in developing their model OhmNet was that tissue layers were in hierarchical order and nodes of tissue layers close to each other would share similar features. I would need more information from the source to verify if this assumption holds true for our data.

Given more time I would want to understand my features and investigate the properties of proteins that might have been ignored in my model.

4. Conclusion

4.1 Applications to Project Management

Reflecting on Zachares et al. 2003, for each construction project task we had a dataset of multiple completion ratios of that same task but no way to relate the task to its neighbourhood of other tasks it was dependent on. In the same way a single protein could exhibit multiple of these functions dependent on the tissue graph network it existed in, so too would a task's completion ratio vary depending on its network with other tasks for a given construction project.

4.2 Summary

It has been a fun exercise which has allowed me to learn about GNNs, understanding the context of the protein-protein interactions data, creating a model and evaluating its performance. I look forward to continue developing and learning about exciting machine learning fields.

References

PapersWithCode. 2023a. *Link prediction on ppi.* Accessed February 7, 2023. https://paperswithcode.com/sota/link-prediction-on-ppi.

———. 2023b. *Node classification on ppi.* Accessed February 7, 2023. https://paperswithcode.com/sota/node-classification-on-ppi.

Zachares, P, V Hovhannisyan, C Ledezma, J Gante, and A Mosca. 2003. On forecasting project activity durations with neural networks. In *International conference on engineering applications of neural networks,* 103–114.

Zitnik, Marinka, and Jure Leskovec. 2017. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics* 33 (14): 190–198.