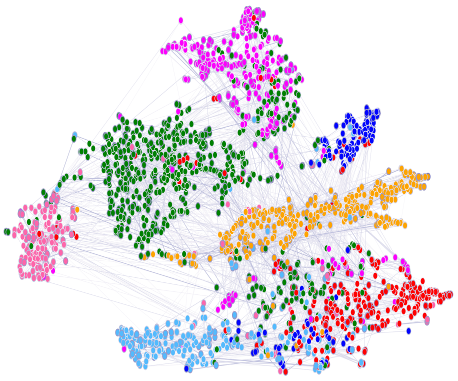
Benchmarking Experiment for Graph Attention Networks

Shreyans Nagori
2018CS10390

Kamalesh Neerasa
2017CS10341

1. INTRODUCTION

Graphical Neural Networks can depict many complex systems such as social, biological and informational networks. At the most abstract level these models are modelled by graphs in which nodes represent individuals or agents and edges represent relationship/interaction between nodes.
Here we benchmark some tests based on biological networks PPI and Protein and on location based networks BrightKite using Graph Attention Networks (Velickovic et Al [1]).
We use Graph Neural networks to perform standard tasks such as pairwise node classification, node classification and link prediction.



: A t-SNE plot of the computed feature representations of a pre-trained GAT model's first hidden layer on the Cora dataset.

2. DATASETS USED

We used 3 datasets for this benchmarking assignment, namely :-
1.**Protein Dataset:** (Borgwardt et al [2])
2.**Protein Protein Interaction:** (Zitnik and Leskovec[3])
3.**Brightkite:** (Cho et al[4])

2.1 PPI Dataset

4 Protein-protein interaction networks from Zitnik and Leskovec. Each graph has 3000 nodes with avg. degree 28.8 and each node has 50 dimensional feature vector. Protein-protein interactions (PPIs) are essential to almost every process in a cell, so understand- ing PPIs is crucial for understanding cell physiology in normal and disease states. It is also essential in drug development, since drugs can affect PPIs. Protein-protein interaction networks (PPIN) are mathematical representations of the physical contacts between proteins in the cell. We use this dataset for the multilabel node classification task and the link prediction task.

2.2 Protein Dataset

This dataset consists of 1113 protein graphs from Borgwardt et al. Each node is labeled with the functional role of the protein. Each node has a 29 dimensional feature vector. We use the this dataset for the pairwise node classification task.

2.3 Brightkite

Brightkite from Cho et al. [2] was once a location-based social networking service provider where users shared their locations by checking-in. The friendship network was collected using their public API, and consists of 58,228 nodes and 214,078 edges. The network is originally directed but they have constructed a network with undirected edges when there is a friendship in both ways. We have also collected a total of 4,491,143 check-ins of these users over the period of Apr. 2008 - Oct. 2010 We use the this dataset for the link prediction task.

	Protein	**PPI**	**Brightkite**
Nodes	43471	56944	58228
Edges	162,088	818,716	214,078
Classes	3	121(Multilabel)	-
Features	29	50	-
Tasks	1	2 and 3	3.

3. PREDICTION TASKS

In this section, we describe in detail the benchmark prediction tasks that we have carried out, as well as the generation of the task-specific dataset. We are using early stopping as a way of regularization, and using scheduler to change learning rate, decaying it by factor of 10 if F1 score or validation loss doesnt improve significantly in the last 10 epoches.

3.1 Pair-wise node classification

Given two nodes, predict if they belong to the same class. This task aims to showcase the similarity of node features / neighbourhoods which influence the class of the node.
We have used the Protein dataset for this task. Our total dataset size was 400,000, which was split for 5-fold cross validation. This was generated by random node-pair sampling equally from the positive samples (both nodes have same class label) and negative samples. The weightage of each class (Class 0, Class 1 Class 2) was proportionate to the number of nodes in that class.

3.2 Link Prediction

Given two nodes, this task predicts if there is an edge between them. In link prediction tasks two nodes are generally more likely to form a link, if they are close together in the graph.
We have used the PPI dataset and the Brightkite dataset for this task. Our total dataset size was 2 times the number of edges in the respective datasets. The negative samples were generated by random sampling of node pairs which don't have edges between them. We used 50 % of edges for purpose of creating embeddings, and divided the remaining edges in a ratio of 3:2 training:testing in first experiment, and in the second experiment we have divided all graphs for 4:1 ratio, and for each set we use 50% edges for creating embeddings, and the remaining edges for testing and training purposes.

3.3 Multi-class node classification

This task involves predicting the class labels for a given node. For the multi-class/ multi-label setting, each node had an associated class label vector. Each node can belong to more than 1 class. We have used the PPI dataset for this task. Each node had an associated 121-dimension binary class label vector, where multiple values can be 1. On average, a node has 37 class labels active (Range 0-101).

4. EXPERIMENTAL SETUP

In this section we outline the scoring methods, implementation details, aggregating functions for both our models and tasks.

4.1 Scoring Methods

ROC AUC: ROC AUC: Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting postive and negative class labels correctly.

Precision: It is the ratio $\frac{TP}{(TP+FP)}$ where TP is the number of true positives and FP the number of false positives. The precision is intuitively the ability of the classifier for successful prediction. The best value is 1 and the worst value is 0.

F1 score: By itself, neither Precision nor Recall are able to capture the classifiers complete performance. So, the F1 score is used to convey the balance between the precision and recall. It is the harmonic mean of Precision and Recall.

4.2 Graph Attention Networks (GATs)

We will start by describing a single graph attentional layer, as the sole layer utilized throughout all of the GAT architectures used in our experiments. The particular attentional setup utilized by us closely follows the work of Bahdanau et al. (2015)—but the framework is agnostic to the particular choice of attention mechanism.

The input to our layer is a set of node features, h = h_1,h_2,...,h_N, $h_i \in R^F$,where N is the number of nodes, and F is the number of features in each node. The layer produces a new set of node features F (of potentially different cardinality than F), h = $h_1^{'}$, $h_2^{'}$, . . . , $h_N^{'}$, $h_i^{'} \in R^{F^{'}}$,as its output.

In order to obtain sufficient expressive power to transform the input features into higher-level features, at least one learnable linear transformation is required. To that end, as an initial step, a shared linear transformation, parametrized by a weight matrix, W $\in R^{F^{'} XF}$, is applied to every node. We then perform self-attention on the nodes—a shared attentional mechanism a : $R^F \times R^F \longrightarrow R$ computes attention coefficients.

That indicate the importance of node j's features to node i. In its most general formulation, the model allows every node to attend on every other node, dropping all structural information. We inject the graph structure into the mechanism by performing masked attention—we only compute e_{ij} for nodes j $\in N_i$. In our experiments, the attention mechanism a is a single-layer feed-forward neural network. Once obtained, the normalized attention coefficients are used to compute a linear combination of the features corresponding to them, to serve as the final output features for every node (after potentially applying a nonlinearity, σ).

4.3 Experiment Setup

For all the tasks in this section, we use 5-fold cross validation for training and evaluation. We report Precision/Recall/F1 scores for Multi-Class Node Classification and ROC AUC scores for Pairwise Node Classification and Link Prediction. For cumulative scores, we report the mean cross-validation scores and test scores corresponding to the best cross-validation score. We consider a range of hyperparameters for tuning in our experiments as described below :
• Layers (**L**) - The number of Graph Attention layers used in our methodology.

• Hidden Dimension (h_{dim})- The dimension of the node embeddings in the hidden layers.
• Hidden GAT Layer Attention Heads (K_{hid}) - The number of attention heads in the hidden layers of GAT network.
• Output GAT Layer Attention Heads (K_{out}) - The number of attention heads in the output layer of GAT network.
• Activation (**act**) - The non linearity activation function used in the GAT network.
• Aggregation (**aggr**) - The aggregation function used in the message passing step. It can be add, max, mean.
• Concatenation for Attention Heads (**concat**): A boolean value indicating whether to concatenate or average the outputs obtained from the different attention heads in the hidden layer.

4.4 Results

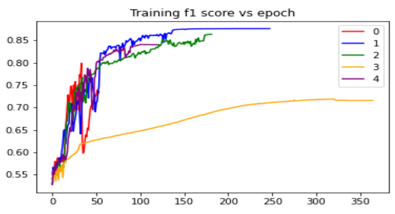
We experimented over the following hyper parameters for each dataset - L $\in$ {2,3,4}, $h_{dim} \in$ {16, 64, 256} and K_{hid} 1,3. A summary of the results for our final model along with tables are given below:

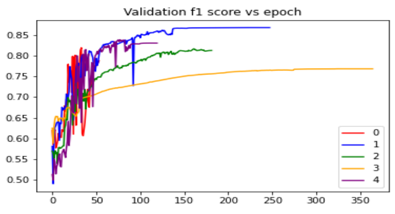
Pairwise Node classification on Protein Dataset:

'Number of Epochs': 1000
'Patience Period': 20
'Number of layers': 2
'Number heads per layer': [1, 1]
'Number features per layer': [29, 64, 64]

Fold Number	Valid ROC AUC	Valid F1 Score	Test ROC AUC	Test F1 Score
1	0.8632	0.8202	0.8210	0.7268
2	0.9407	0.8690	0.9475	0.8807
3	0.8632	0.8202	0.9699	0.9085
4	0.8288	0.7688	0.7989	0.7523
5	0.8948	0.8312	0.9426	0.8753
Cumulative	0.8781	0.8218	0.8959	0.8287

Results of ROC-AUC and F1 scores over all k=5 Folds over pairwise node classification

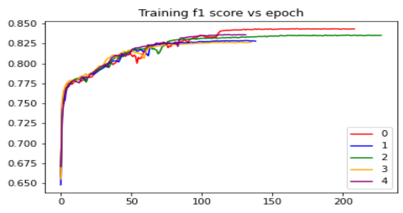


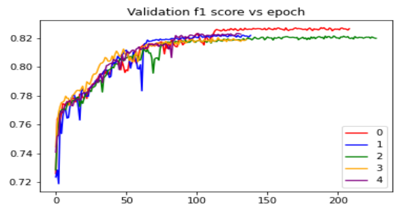


Link Prediction on PPI Dataset:

Number Of Layers: 2
Number Heads Per Layer : [4, 4]
Number Features Per Layer : [50, 64, 64]

Fold Number	Valid ROC-AUC	Valid F1-Score	Test ROC-AUC	Test F1-Scor
1	0.9058	0.8274	0.9047	0.8254
2	0.9014	0.8222	0.8959	0.8158
3	0.9018	0.8219	0.9018	0.8219
4	0.8991	0.8204	0.9000	0.8196
5	0.9019	0.8234	0.9065	0.8271
Cumulative	0.902	0.8238	0.9017	0.8219

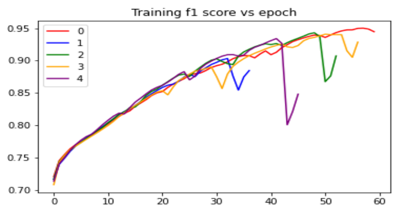


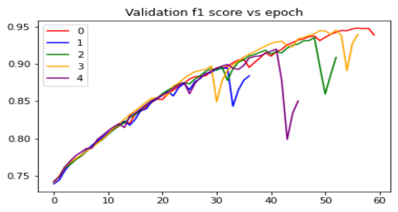


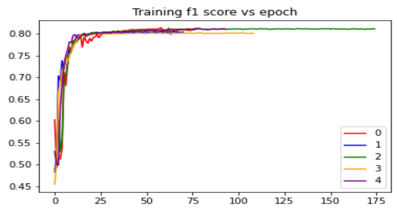
BrightKite Dataset Link Prediction

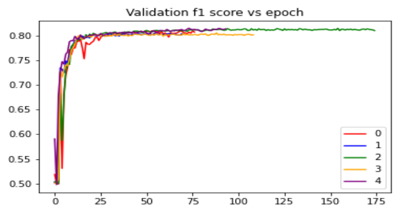
Number of epochs: 1000
patience period: 20
Number of layers: 3
Number heads per layer: [4, 4, 4]
Number of features per layer': [features, 64, 64, 64]

Fold Number	Valid ROC-AUC	Valid F1-Score	Test ROC-AUC	Test F1-Score
1	0.8898	0.8102	0.8863	0.8026
2	0.8846	0.8036	0.8785	0.7973
3	0.8943	0.8103	0.8922	0.8096
4	0.8816	0.8018	0.8855	0.8062
5	0.8971	0.8140	0.9014	0.8164
Cumulative	0.8884	0.8079	0.8887	0.8052









PPI Dataset Multilabel Node Classification

Fold Number	Valid ROC-AUC	Valid F1-Score	Test ROC-AUC	Test F1-Score
1	0.9805	0.9389	0.9809	0.9393
2	0.9412	0.8840	0.9514	0.8949
3	0.9603	0.9354	0.9738	0.9320
4	0.9814	0.9399	0.9796	0.9354
5	0.9051	0.8502	0.9683	0.9242
Cumulative	0.9537	0.909	0.9708	0.9251

4.5 Additional experiments

We also checked variation of test F1 and ROC with changing hidden layer size in brightkite for 1 head and 3 layers of the same size and we got the following results:

Hidden Size	Test F1	ROC
16	0.8039	0.8885
32	0.8049	0.8897
64	0.8052	0.8888
128	0.8084	0.8919

Now varying no of heads and keeping no of tables fixed.

Num Heads	Test F1	ROC
1	0.8039	0.8885
2	0.8127	0.8968
4	0.8127	0.8975
8	0.8133	0.8956

Now keeping numheads fixed at 1 and varying hidden layers no with size as 16.

Layers	Test F1	ROC
1	0.670	0.6907
2	0.783	0.8609
3	0.8146	0.8975

5. CONCLUSIONS

Thus, to conclude, we have investigated the applications of Graph Attention Networks on Link Prediction and Node Classification tasks. We observed that other than Pairwise Node Classification, GAT networks are pretty much state of the art in tasks like Link Prediction and Node Classification. We also observed that GATs reach their optimal performance very quickly for small datasets, however, for large graphs GATs need much time to achieve SOTA performance. Further, we analysed the hyperparameters used in our experiments and observed that higher number of layers and lower dimensions in hidden layer can boost performance in smaller datasets. Moreover, we observed that having multiple attention heads is more useful for outer GAT layers rather than inner layers. It was also observed that the tanh activation can significantly boost the quality of our predictions. Finally we make our code public so as to facilitate further understanding of embeddings generated by GAT networks.

Code is available at https://github.com/whywasievenhere/COL-868-GNN-Benchmarking .