GML-FA Assignment-3 report

Aayush Sugandh
24AI60R04

Sandipan Majhi
24AI60R21

1 Methodology

In this assignment we worked how Graph Convolutional Network (Kipf and Welling, 2016) combined with GPool Downsampling layer (Gao and Ji, 2019) and DiffPool layer (Ying et al., 2018) perform on Enzymes dataset and D&D Dataset (Morris et al., 2020). We performed all the experiments using Pytorch library.

1.1 Dataset

The information about the datasets can be found in Table 1. We observe that Enzymes dataset has smaller graphs which are only 600 in number but 6 classes. It is expected that deeper models might show overfitting and low performance on test set. On the hand, D&D dataset has larger number graphs which have larger number of graphs. So the training was much more stabilized and had lesser overfitting.

1.2 Batching

In order to run our experiments on GPUs we had to batch our adjacency matrices and feature matrices. The main problem was that, different graphs have different number of nodes. So, we found out maximum number of nodes for a batch of graphs and augmented other graphs in the batch with zeros in adjacency matrices and feature matrices. Then we pass these batches into our models.

1.3 Model Implementation

For GCN we implemented (Kipf and Welling, 2016). In addition to stabilize training and reduce overfitting as much as possible, we applied BatchNorm and Normalization to the GCN output before applying ReLU activation.

For Gpool layer we implemented similar to (Gao and Ji, 2019) and for DiffPool layer we implemented (Ying et al., 2018).

In addition to suggestion in the assignment, $GCN_1 \rightarrow GCN_2 \rightarrow GPool_1 + DiffPool_1 \rightarrow GCN_3 \rightarrow GCN_4 \rightarrow Classifier$. We employ **skip connections** as suggested by (Kipf and Welling, 2016) between outputs of GCN_1 to output of GCN_2 and from GCN_3 to output of GCN_4. Also we employed **layernorm** to this skip connection output.

The model is presented in figure - 3.

Dataset	# Graphs	Mean Nodes	Classes
Enzymes	600	32.63	6
D&D	1178	284.32	2

Table 1: Datasets on which experiments were performed

2 Hyper Parameters

We used $Lr = 0.005$, $Weight\_Decay = 5 * 10^{-4}$ for our training. For Enzymes dataset, we used Batch size of 256 and for D&D Dataset we use Batch Size of 8. The embedding dimension for all our GNNs including downsample and pooling layers we used, 32. Also before classification head we used a dropout layer with probability 0.1.

k_1	k_2	m_1	m_2	Mean Test Acc ± Stdev Test Acc	Max Test Acc	Max Test Acc in 50 Epochs
0.9	0.9	6	3	0.7508 ± 0.0166	0.7712	0.7966
0.9	0.8	6	3	0.7559 ± 0.0143	0.7797	0.8051
0.9	0.6	6	3	0.7492 ± 0.0099	0.7712	0.7881
0.8	0.9	6	3	**0.7602** ± **0.0133**	**0.7797**	**0.8051**
0.8	0.8	6	3	0.7475 ± 0.0119	0.7627	0.7881
0.8	0.6	6	3	0.7720 ± 0.0238	0.8051	0.8051
0.6	0.9	6	3	0.7585 ± 0.0151	0.7966	0.7966
0.6	0.8	6	3	0.7297 ± 0.0185	0.7627	0.7881
0.6	0.6	6	3	0.7585 ± 0.0115	0.7712	0.8051

Table 2: Test Accuracy Results for D&D Dataset over 50 epochs.

k_1	k_2	m_1	m_2	Mean Test Acc ± Stdev Test Acc	Max Test Acc	Max Test Acc in 1000 Epochs
0.9	0.9	6	3	0.5500 ± 0.0283	0.5833	0.6333
0.9	0.8	6	3	0.4783 ± 0.0452	0.5833	0.6000
0.9	0.6	6	3	0.4500 ± 0.0283	0.4833	0.5333
0.8	0.9	6	3	**0.5533** ± **0.0358**	**0.6167**	**0.6667**
0.8	0.8	6	3	0.3783 ± 0.0401	0.4333	0.5167
0.8	0.6	6	3	0.4567 ± 0.0326	0.5000	0.6000
0.6	0.9	6	3	0.3900 ± 0.0654	0.4667	0.5333
0.6	0.8	6	3	0.2583 ± 0.0403	0.3167	0.4167
0.6	0.6	6	3	0.4867 ± 0.0450	0.5500	0.5667

Table 3: Test Accuracy Results for Enzymes Dataset over 1000 Epochs

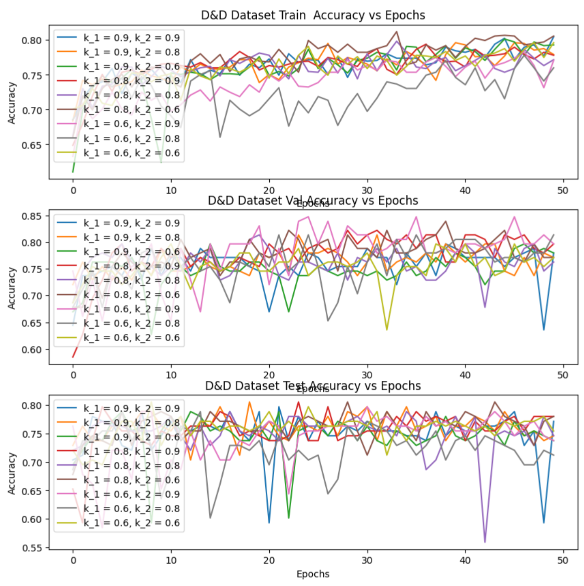

Figure 1: Training on D&D Dataset.

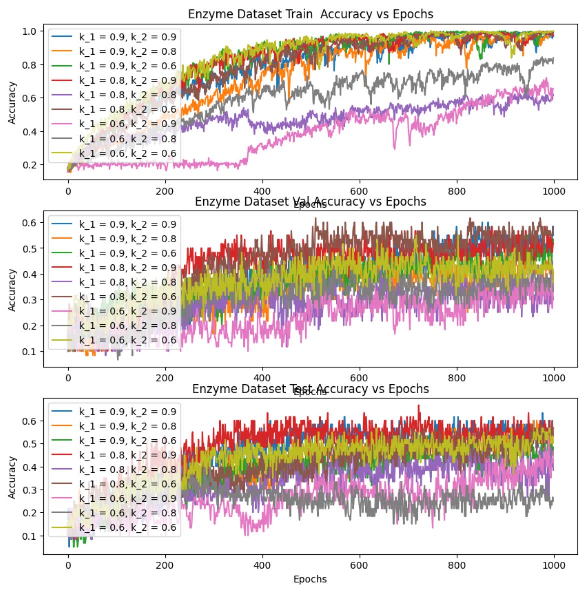

Figure 2: Training on Enzyme Dataset

3 Findings and Discussion

From our experiments we find that, the best setting of the models for this task was $k_1 = 0.8$,$k_2 = 0.9$, $m = 6$ and $m = 3$. The test statistics presented in tables (2) for D&D dataset and table (3) for Enzymes dataset are calculated from the top 10 models in all epochs trained based on higher validation accuracy. Then we calculated the mean and standard deviation of the test accuracies for these top 10 models and reported here. Also we observed that when $k_1 < k_2$, the model performs worse. This could be because, during downsampling when it is too much downsampled it loses too much information for inference.

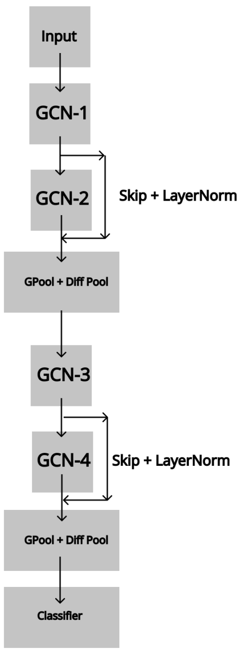

Figure 3: Model Diagram.

Also during training of Enzymes dataset, we found that there is significant overfitting problem. It is probable that because we had very less training data and graphs were also smaller was the main reason for this behaviour. Also this problem was harder to solve as we had 6 classes to predict for each graph. In this case, probably some unsupervised pre-training or transfer learning could be helpful in getting higher accuracy. We also find significant noisy training over this dataset where there were frequent fluctuations in training and validation metrics. Based on the top 10 validation accuracy data, we got the test accuracy results for each setting, which are presented in table (3). According to that, we found that $k_1 = 0.8$,$k_2 = 0.9$, $m = 6$ and $m = 3$ with mean test accuracy of $\mathbf{0.5533 \pm 0.0358}$ and maximum test accuracy achieved by these top 10 models were, **0.6167**. The training, validation accuracies over epochs are shown in figure - 2.

For D&D Dataset, we find that for batch size 8 (due to GPU memory restrictions), training and validation accuracies were not changing much. So we restricted our training to only 50 epochs in this case. Just like Enzymes dataset, we also got, $k_1 = 0.8$,$k_2 = 0.9$, $m = 6$ and $m = 3$ as best model with mean test accuracy $\mathbf{0.7602 \pm 0.0133}$ and maximum test accuracy of **0.7797**. Here also we got the test accuracy after evaluating over top

10 models based on higher validation set accuracy. This dataset had larger graphs, so in this case, there were lesser overfitting problem but interestingly we found some problems with train accuracy getting stuck.

References

Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:4948–4960.

Thomas Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*.

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. *ArXiv*, abs/1806.08804.