ECS 189G-001
Deep Learning

Winter 2023

Course Project: Stage 5 Report

Team Information

Enter Your Team Name Here (delete the extra rows if your team has less than 4 students)		
Name : Chung Ying Hsu	ID: 920918764	Email: cyhsu@ucdavis.edu
Name: Shuhao Gao	ID: 919412501	Email: shgao@ucdavis.edu

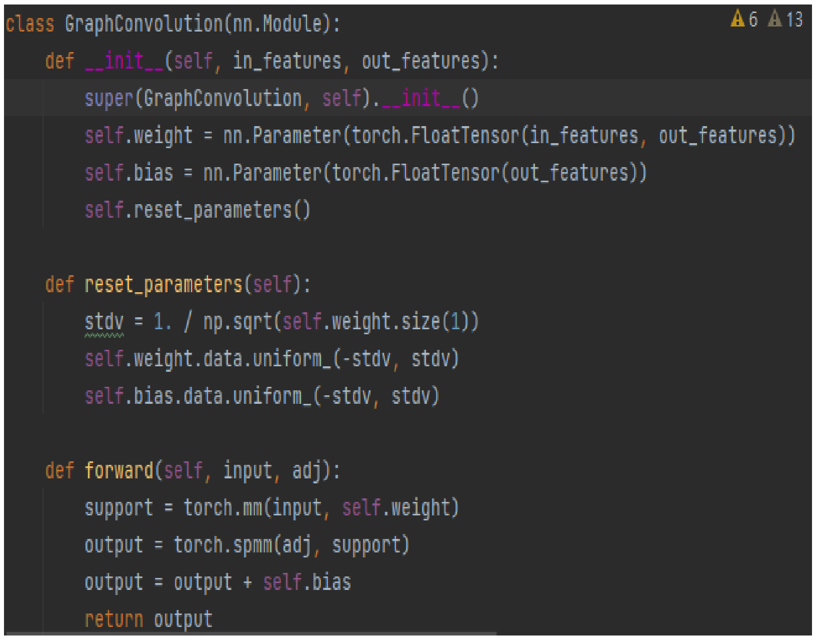
Section 1: Task Description

In this project, we investigate the Graph Neural Network (GNN) model, and use the model to embed graphs and classify nodes by Cora, Pubmed and Citeseer datasets.During the training process, we need to ensure that the accuracy of Cora, Pubmed reach 70% and Citeseer should be above 60%.

Section 2: Model Description

In GraphConv(GCN), the input vector is first multiplied by the weight matrix to obtain the support matrix, and then the adjacency matrix adj is multiplied by the support matrix through sparse matrix multiplication to obtain the final output vector. Finally, the output vector is added to the bias vector to obtain the final output feature vector.

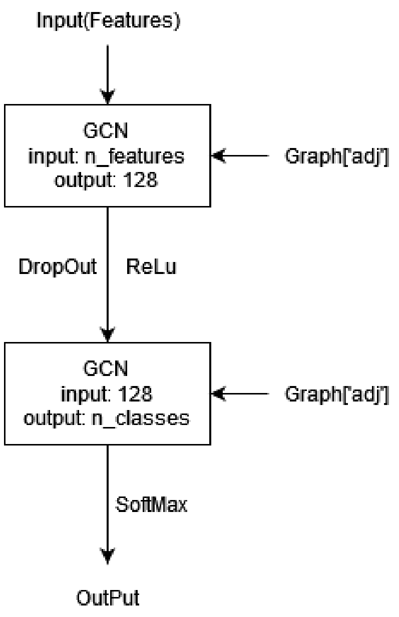
Input the features and adjacency matrix into the first layer of GCN, output with dim=128, and then input these 128 features and adjacency matrix into the second layer of GCN. The second layer of GCN outputs with dim = n_classes. Then let the output go through softmax processing to get the prediction result of which class the input is.



```
class GraphConvolution(nn.Module):
    def __init__(self, in_features, out_features):
        super(GraphConvolution, self).__init__()
        self.weight = nn.Parameter(torch.FloatTensor(in_features, out_features))
        self.bias = nn.Parameter(torch.FloatTensor(out_features))
        self.reset_parameters()

    def reset_parameters(self):
        stdv = 1. / np.sqrt(self.weight.size(1))
        self.weight.data.uniform_(-stdv, stdv)
        self.bias.data.uniform_(-stdv, stdv)

    def forward(self, input, adj):
        support = torch.mm(input, self.weight)
        output = torch.spmm(adj, support)
        output = output + self.bias
        return output
```



Section 3: Experiment Settings

3.1 Dataset Description

Cora:

Cora is a graph dataset for paper classification, where each document is represented as a node. The Cora dataset consists of 2708 research papers belonging to 7 different disciplines. Divide papers into the following categories: Mathematics, Physics, Computer Science, Ergonomics, Biology, Chemistry, and Medicine.

2708 Nodes, 7 classes, 5429 directed edges.

Citeseer:

Citeseer is a dataset for academic papers, where each document is represented as a node. The Citeseer dataset consists of 3312 research papers belonging to 6 different disciplines. Divide papers into the following categories: AI, Agents, DB, HCI, IR, ML.

3312 Nodes, 6 classes, 4715 directed edges.

Pubmed:

PubMed is a graph dataset for paper classification, where each document is represented as a node. The PubMed dataset consists of 19,717 medical papers belonging to 3 different disciplines. Divide papers into the following categories: Diabetes, Kidney, Liver.

19717 Nodes, 3 classes, 44324 directed edges.

Due to the instructions in the task, 20 nodes of each category are randomly selected. Therefore, after I classify the node types and fix the random seed, I use random.shuffle to reorder the nodes. And make trainset, testset choose the required number from it. In this way, the trainset and testset can meet the requirements of the task.

3.2 Detailed Experimental Setups

All Dataset:

Depth: 2 * GCN
Weight Decay: 5e-4
Learning rate: 0.01
Epoch: 200
Batch: Full Batch
Loss function: CrossEntropyLoss (This task is multi-classification, so I choose CrossEntropyLoss as my loss function.)
Opimizer: Adam

Cora:

Layer dimensions: features_n(1433) -> 128 -> classes_n(7)

Citeseer:

Layer dimensions: features_n(3703) -> 128 -> classes_n(6)

Pubmed:

Layer dimensions: features_n(500) -> 128 -> classes_n(3)

3.3 Evaluation Metrics

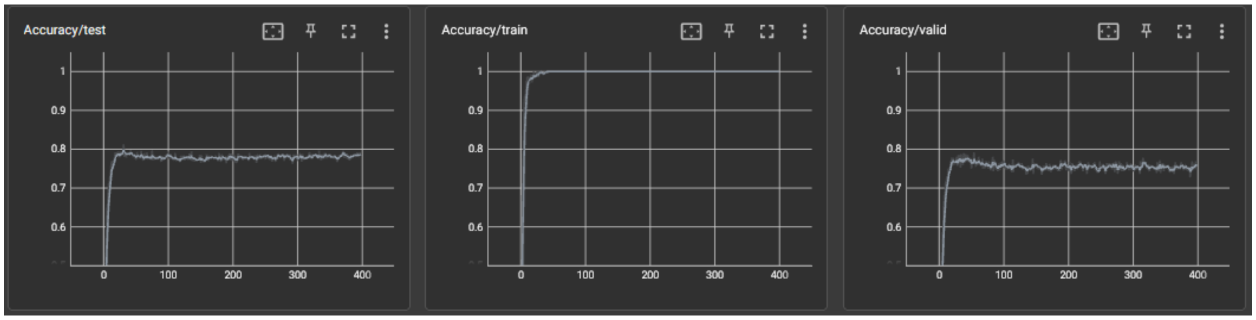
I will use Accuracy, Precision, Recall, F1 score as our evaluation metrics.

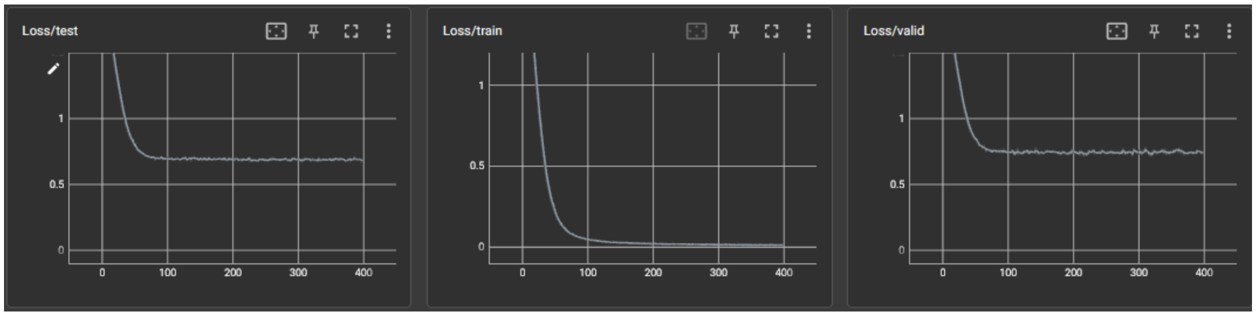
3.4 Source Code

https://drive.google.com/file/d/19wisAPUZA9dLA-sAtDuqpLuYAxXdldKn/view

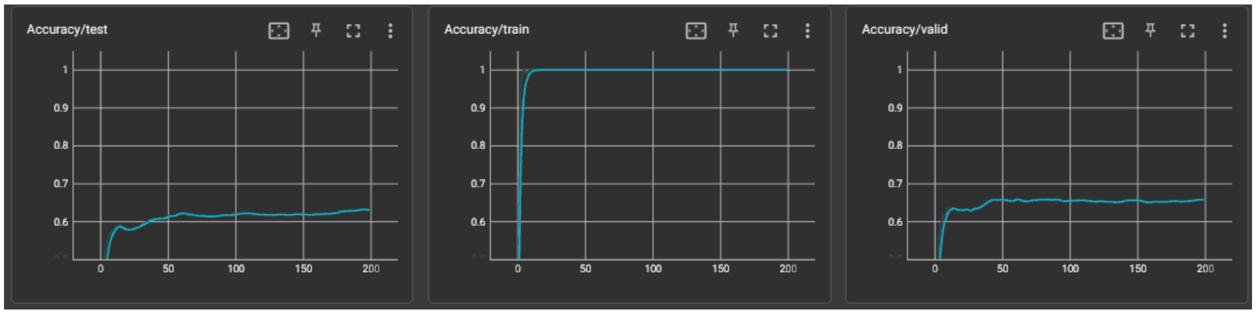
3.5 Training Convergence Plot

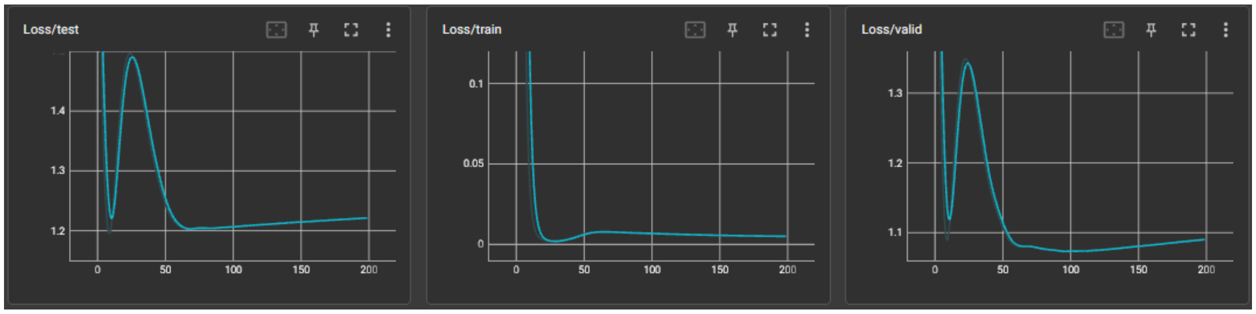
Cora:



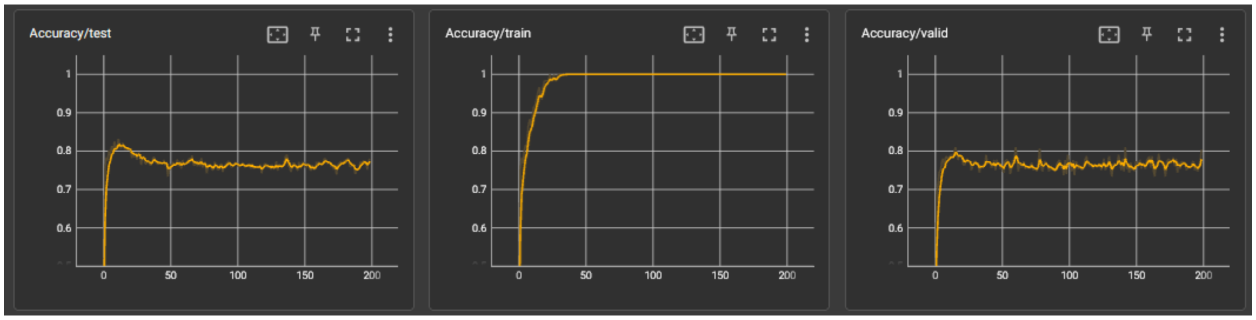


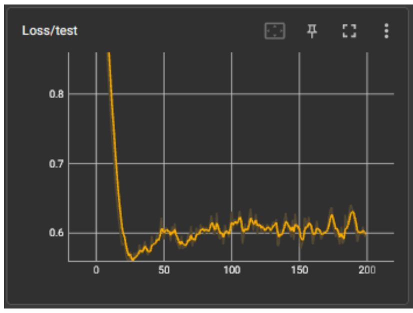
Citeseer:

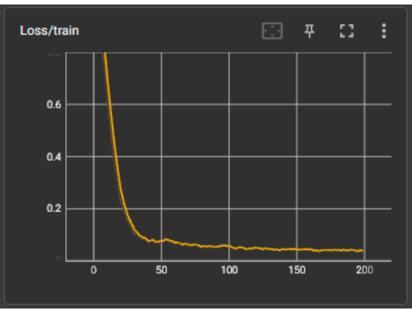


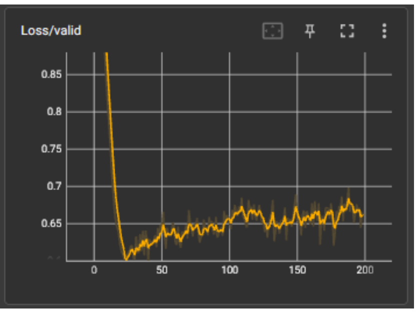


Pubmed:









It can be clearly seen here that the accuracy rate and loss are gradually converging.
There is an interesting fact. In pubmed and cora, when there are 20 epochs, the accuracy of Cora and Pubmed will be more that 80%. However, when the epoch = 20, the loss doesn’t converge, so we don’t think max_epoch=20 is a good idea.

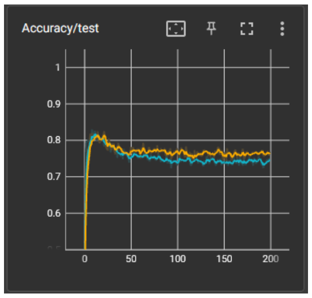
3.6 Model Performance

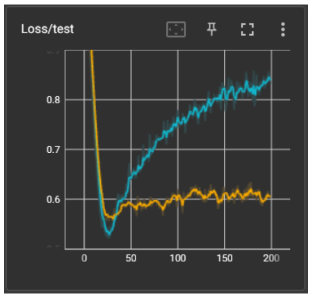
Score	Cora		Citeseer		Pubmed	
	Train	Test	Train	Test	Train	Test
Accuracy	1	0.78476	1	0.63	1	0.775
Precision	1	0.79643	1	0.6308390	1	0.7750443
Recall	1	0.78476	1	0.63	1	0.775
F1	1	0.783393	1	0.619469	1	0.774720

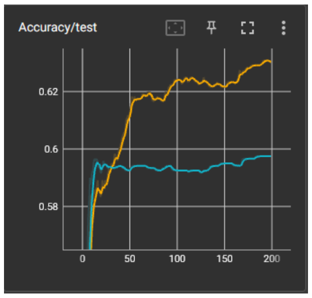
3.7 Ablation Studies

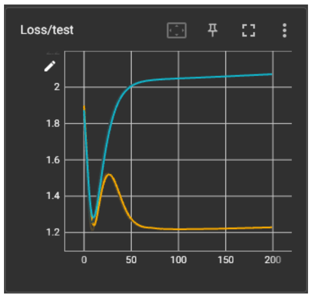
1. Whether or not weight-decay is used has a great influence on the model. The following is the same structure, but does not use weight-decay for regularized learning curve. The model without regularization will only perform better in the training set. In the test set and verification set, the performance will be very poor, and it is very easy to overfit. (Blue line is model without weight-decay)

Pubmed | Citeseer

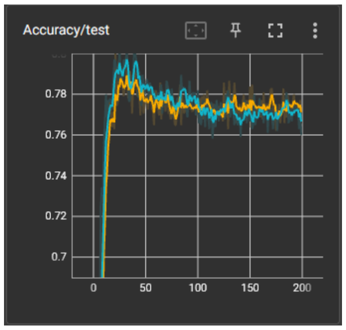


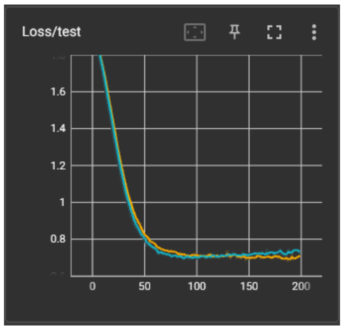




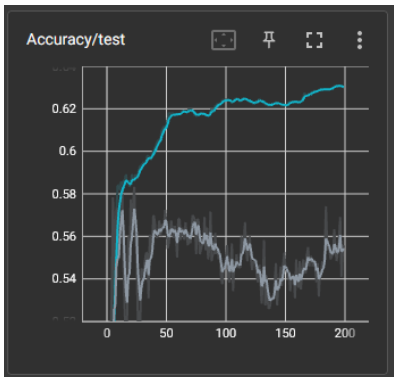


Cora doesn't have much noticeable difference.





2. I tried using 4 layers of GraphConv to train the models, and there is not much difference in their performance on Cora for 200 epochs. But in citeseer, you can see that there are obvious differences.

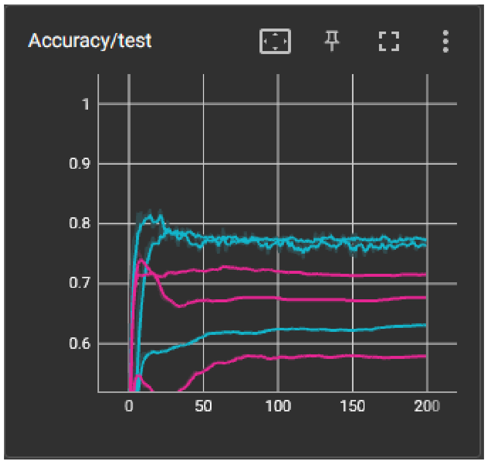


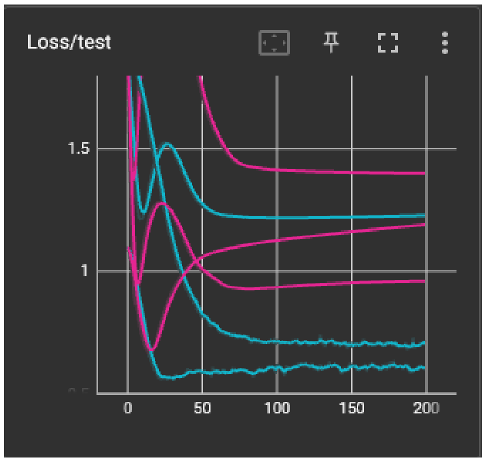
gray line is 4 layer model.

I found that if you want to do a higher number of layers of GraphConv, we need to do a similar operation to the CNN, we need to use the Residual Conv layer. But I didn't build a useful ResBlock, so I didn't test it.

3.

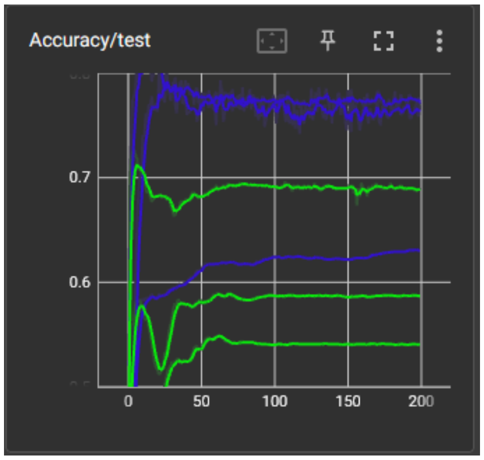
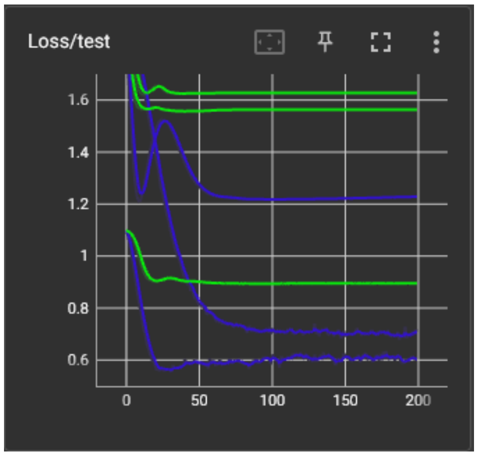
The blue line is using GraphConv created by ourselves, while the red line is the model created using GCNConv and GATConv in torch_geometric.nn. The hyperparameters are all the same. It can be seen that the model created with torch_geometric.nn does not perform very well.





4.

The green line is the learning curve of the MLP model. The blue line is our own GraphConv. The effect of MLP is worse than that of using torch_geometric.nn.


Accuracy/test

Loss/test