Name: [Shusmita Biswas]

University: [Harbin Institute of Technology]

Student ID: [24SF51033]

Experiment 3: PyTorch Implementation of Graph Convolutional Neural Network Classification

# 1. Introduction

Graph Convolutional Networks (GCNs) are a class of neural networks designed to operate on graph-structured data. They extend the concept of convolution from images to graphs by aggregating and transforming feature information from neighboring nodes. In this experiment, we explore the use of GCNs for node classification tasks using citation network datasets such as Cora and PubMed, and later enhance the model using Graph Attention Networks (GATs) to achieve better accuracy.

# 2. Environment Setup

To run this experiment smoothly, we followed a structured environment setup using Anaconda:

## Step 1: Install Anaconda

• Download Anaconda from the official site: https://www.anaconda.com/products/distribution  
• Install it by following the OS-specific installation instructions.

## Step 2: Create a Virtual Environment

We created a dedicated virtual environment for this experiment to avoid conflicts:

conda create --name experiment\_3 python=3.9  
conda activate experiment\_3

## Step 3: Install Necessary Packages

Inside the experiment\_3 environment, install required packages:

A screenshot of a computer

AI-generated content may be incorrect.

pip install torch torchvision torchaudio  
pip install torch-geometric  
pip install scikit-learn matplotlib

# 3. Initial GCN Model on Cora Dataset

Dataset: Cora  
Model Used: Two-layer GCN  
Library: torch\_geometric  
Accuracy Achieved: 77.9%

We first implemented a simple Graph Convolutional Network (GCN) using the Cora dataset, a benchmark dataset for citation network classification. The model consists of two GCNConv layers and uses ReLU activation, Dropout, and Adam optimizer.

Key Code Snippet:  
self.conv1 = GCNConv(in\_channels, hidden\_channels)  
self.conv2 = GCNConv(hidden\_channels, class\_n)

A screenshot of a computer program

AI-generated content may be incorrect.

Observations:  
• The model was trained for 10 epochs.  
• Final accuracy reached 77.9% on the test set.

# 4. GCN Model on PubMed Dataset

Dataset: PubMed  
Model Used: Same two-layer GCN  
Accuracy Achieved: 71.8%

To test the generalizability of the model, we applied the same GCN architecture on the PubMed dataset. This dataset has different properties and node distributions compared to Cora.

Code Differences:  
data = Planetoid(root='./data', name='PubMed')[0]

A screenshot of a computer program

AI-generated content may be incorrect.

Observations:  
• Performance dropped to 71.8%, likely due to differences in graph sparsity and feature characteristics.  
• This highlighted the limitation of plain GCNs in learning complex relationships in graph data.

# 5. Improved Model using GAT on Cora Dataset

Dataset: Cora  
Model Used: 3-layer Graph Attention Network (GAT)  
Accuracy Achieved: 80.4%

To improve performance, we implemented a GAT (Graph Attention Network) which uses attention mechanisms to weigh the importance of neighbor nodes.

Key Improvements:  
• Switched from GCNConv to GATConv  
• Introduced multi-head attention (8 heads) in the first layer  
• Used 3 layers for deeper feature learning  
• Added residual connections and dropout  
• Applied early stopping based on validation accuracy

Model Highlights:  
self.gat1 = GATConv(in\_channels, hidden\_channels, heads=8, dropout=0.6)  
self.gat2 = GATConv(hidden\_channels \* 8, hidden\_channels, heads=1, concat=False)  
self.gat3 = GATConv(hidden\_channels, out\_channels, heads=1, concat=False)

Training Strategy:  
• 200 epochs with early stopping  
• Learning rate: 0.002  
• Dropout rate: 0.6  
• Weight decay: 1e-4

Results:

A screenshot of a computer program

AI-generated content may be incorrect.  
• Best test accuracy reached 80.4%  
• Early stopping improved training efficiency  
• The attention mechanism allowed the model to better capture relevant node dependencies

# 6. Conclusion

Through progressive experimentation:  
• We started with a simple GCN and achieved 77.9% on Cora.  
• Applying the same on PubMed resulted in lower accuracy (71.8%).  
• Finally, switching to GAT and adding attention, deeper layers, and dropout improved performance to 80.4%.  
  
This shows that model architecture, dataset characteristics, and training strategies significantly impact the effectiveness of GNNs in node classification tasks.