Programming Assignment 1: GNN-based Node Classification on Cora

Summary of Findings for Graph Learning (2025W)

Simon Schindler, Marlene Grabner, Aryan Rahbari

December 15, 2025

1 Introduction

This report summarizes the findings for the semi-supervised node classification tasks on the Cora citation network, as required by Programming Assignment 1. We utilized the PyTorch Geometric library for all experiments.

Python Environment Setup

To run the Jupyter notebooks and reproduce the results, a specific Python environment is required. The following steps outline the setup process using `uv`, a fast Python package installer.

1. **Install `uv`:** If not already installed, run the following command:

   ```
   curl -Lsf https://astral.sh/uv/install.sh | sh
   ```

2. **Create Virtual Environment:** This project uses Python 3.13. From the project root, create a virtual environment:

   ```
   uv venv
   ```

   `uv` will automatically detect the required Python version from the `.python-version` file.

3. **Install Dependencies:** Activate the environment and install packages using the `sync` command:

   ```
   uv pip sync
   ```

   This command uses the `pyproject.toml` and `uv.lock` files to ensure a reproducible environment. You can then start Jupyter Lab by running `jupyter lab`.

2 Task 2(a): Hyperparameter Optimization and Best Model

The objective of this task was to find a GCN-based model that achieves at least 80% test accuracy on the Cora dataset using the standard validation split. A comprehensive grid search was performed over key hyperparameters, including hidden dimensions, number of layers, dropout rate, learning rate (LR), and convolutional layer type (GCNConv, GraphConv, GATConv).

Table 1 summarizes the hyperparameters evaluated.

Hyperparameter	**Evaluated Values**
Hidden Dimensions	**16**, 64, 128
Number of Layers	**2**, 4, 8
Dropout	0.3, **0.5**, 0.7
Learning Rate	0.01, 0.005, **0.001**
Epochs	10, 50, **200**
Convolution Layer	**GCNConv**, GraphConv, GATConv

Table 1: Hyperparameters evaluated in the grid search. The best configuration is marked in **bold**.

2.1 Optimal GNN Architecture

The best-performing model, selected based on the highest validation accuracy, significantly exceeded the target, achieving a Test Accuracy of 88.50%. The optimal hyperparameters and resulting performance metrics are summarized in Table 2.

Table 2: Optimal GCN Architecture and Performance on Cora

Hyperparameter	**Setting**	**Notes**
Convolutional Layer	`GCNConv`	Standard GCN
Number of Layers	2	Shallow architecture
Hidden Channels	16	Smallest dimension tested
Dropout Rate	0.5	Optimal regularization
Learning Rate (LR)	0.001	Slow, stable convergence
Epochs	200	Full training run
Performance	**Accuracy**	
Training Accuracy	95.36%	
Validation Accuracy	88.00%	(Used for selection)
Test Accuracy	88.50%	(> **80**% target reached)

2.2 Key Hyperparameter Influences

Analysis of the grid search results revealed clear patterns regarding model performance and overfitting:

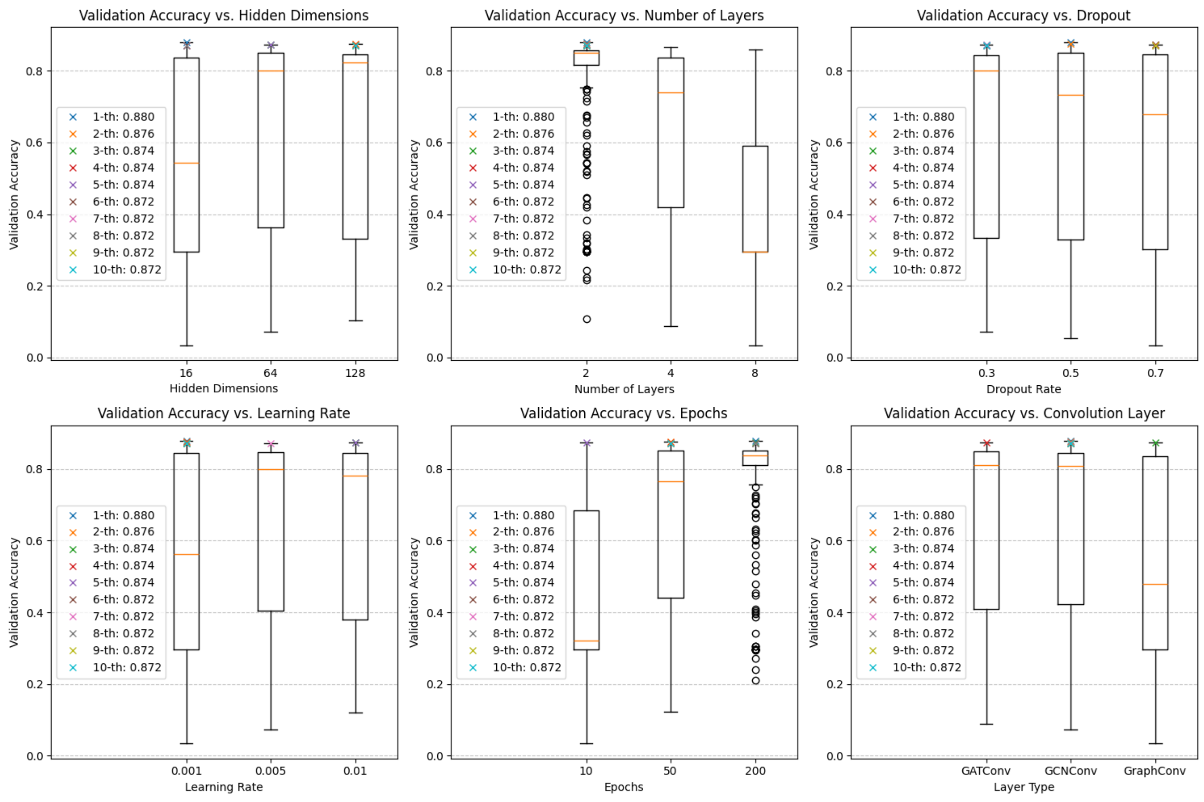

Figure 1: Influence of model hyperparameters on validation accuracy. Best 10 runs are individually marked with crosses.

Analyzing fig. 1 yields the following insights:

- **Number of Layers:** Performance peaked dramatically at 2 layers. Deeper models (4 and 8 layers) showed a significant drop in accuracy, indicating that the standard GCN architecture quickly suffers from **oversmoothing** on this dataset, where node representations become indistinguishable across the graph.
- **Hidden Dimensions:** The smallest dimension tested, 16, was optimal. Models with 64 or 128 channels showed decreased accuracy and a greater tendency toward overfitting (larger gap between training and validation accuracy).
- **Learning Rate:** A low LR of 0.001 was crucial for achieving the highest accuracy, suggesting that fine-tuning weights over many steps (200 epochs) is more effective than rapid learning.
- **Layer Type:** The standard `GCNConv` performed marginally better than `GraphConv` and significantly better than `GATConv`, suggesting the attention mechanism was not necessary for performance improvement in this setting.

3 Task 2(b): Homophily Analysis

This task focused on analyzing the graph's homophily and investigating the hypothesis that nodes with low homophily scores are more likely to be misclassified by the optimized GNN.

3.1 Homophily Measurement

3.1.1 Overall Homophily

The overall Node Homophily H_{node} was calculated using the formula:

$$H_{\text{node}} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{|\{(w, v) : w \in \mathcal{N}(v) \wedge y_v = y_w\}|}{|\mathcal{N}(v)|}$$

The computed homophily score for the Cora dataset is $\mathbf{H}_{\text{node}} \approx \mathbf{0.8252}$. This confirms that Cora is a **highly homophilous graph**, where approximately 82.5% of a node's neighbors share its class label.

3.1.2 Homophily Distribution

Figure /reffig:homohist visualizes a histogram of per-node homophily scores showing a severely skewed distribution, with the vast majority of nodes (over 65%) residing in the highest homophily bin $(0.9 - 1.0)$, further solidifying the homophilous nature of the graph.

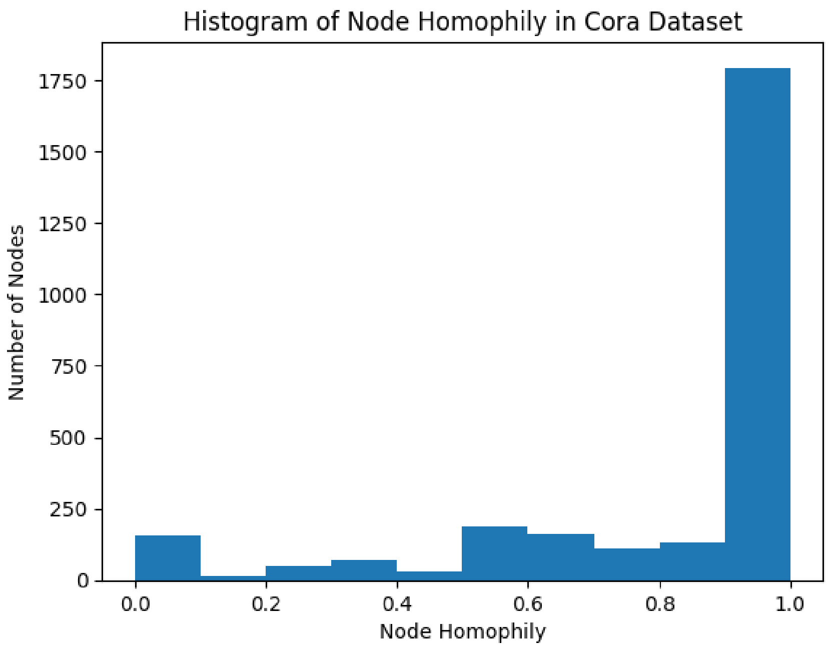

Figure 2: Homophily Distribution in Cora

3.2 Predictive Performance vs. Homophily

3.2.1 Experimental Design

To test the relationship between homophily and predictive performance, the nodes in the validation set were partitioned into 10 bins based on their individual homophily score (e.g., $0.0 - 0.1$, $0.1 - 0.2$, ..., $0.9 - 1.0$). The classification accuracy of the top 10 models (identified in Task 2a) was then measured for the nodes within each bin.

3.2.2 Findings

The results, visualized in a box plot (Figure 3), provide a clear answer to the central question:

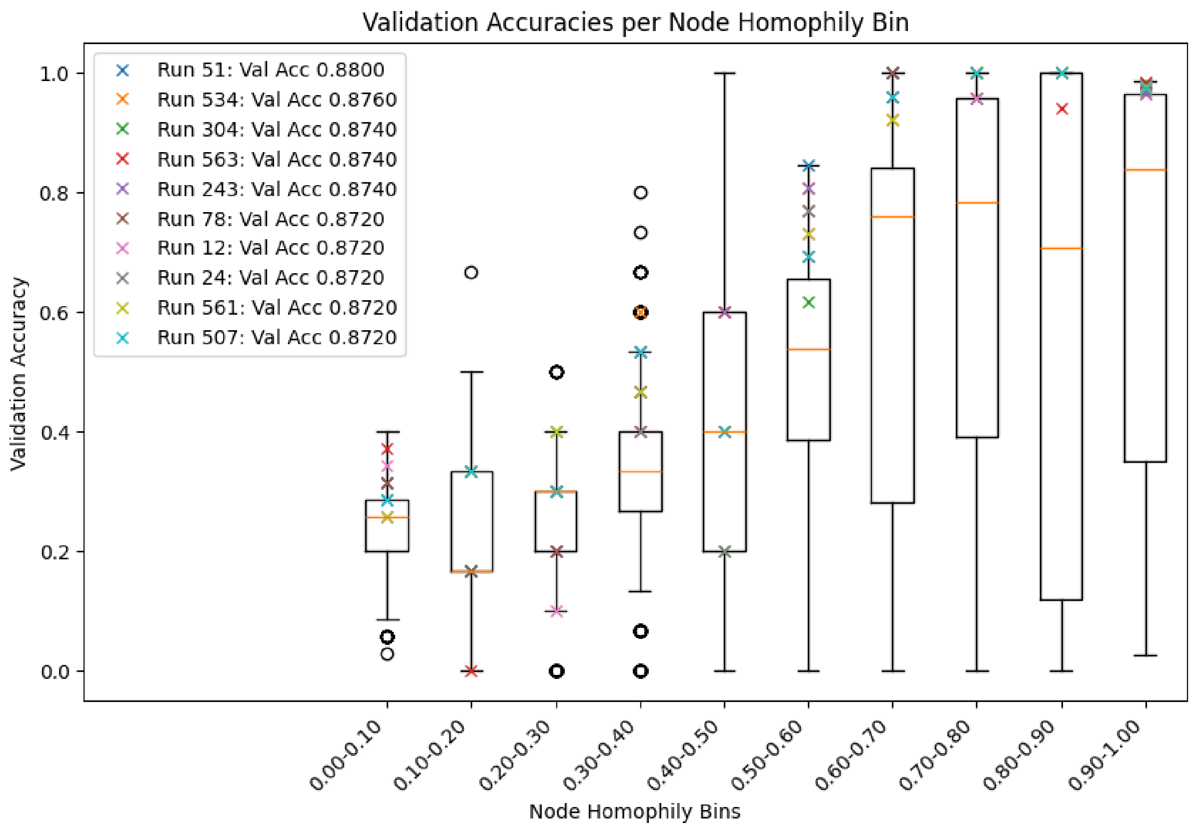

Figure 3: Validation Accuracy Distribution Across Node Homophily Bins

- **High Homophily (≥ 0.8):** Models show **excellent performance**. In the $0.9 - 1.0$ bin, the median accuracy across all runs approached 100%. The GCN performs optimally when the neighborhood signal is consistent.
- **Low Homophily (< 0.5):** The models consistently struggle. The median accuracy in the $0.0 - 0.1$ bin is very low (well below 20%), indicating that the majority of nodes in this bin are misclassified.
- **Conclusion:** Nodes with low homophily are **significantly more likely to be misclassified**. This is a characteristic failure mode of standard GCNs, which operate under the assumption of homophily. When a node is heterophilous (linked to different classes), the message-passing mechanism aggregates conflicting labels, corrupting the node's representation and leading to incorrect predictions.