Understanding Graph Neural Networks: Motivation, Models, and Empirical Evaluation

Merhawit Tesfay Kassa

December 17, 2025

1 Introduction

Many real-world datasets are inherently **relational**, meaning that the behavior or label of one entity depends on its relationships with other entities. Examples include social networks, molecular structures, recommendation systems, and citation networks. In such cases, representing data as independent samples leads to a loss of crucial structural information.

Traditional machine learning models typically assume that data points are **independent and identically distributed (i.i.d.)** and represented in tabular form. Models such as linear classifiers or multi-layer perceptrons (MLPs) process each sample independently and therefore struggle to exploit relationships between entities. This limitation is analogous to using an MLP for image recognition without convolutional neural networks (CNNs), which fail to capture spatial locality.

Graph Neural Networks (GNNs) address this challenge by explicitly modeling relationships between entities. In this work, several GNN architectures are evaluated on the Cora citation dataset to understand how different graph-based aggregation mechanisms affect node classification performance.

2 What is a Graph Neural Network (GNN)

A Graph Neural Network is a neural network designed to operate on graph-structured data. A graph is defined as $G = (V, E)$, where V represents nodes and E represents edges. Each node may have associated features, and edges encode relationships between nodes.

GNNs learn node representations by iteratively aggregating information from neighboring nodes. This process, known as **message passing**, allows nodes to exchange information and produce representations that capture both node attributes and graph structure.

3 Why Graph Structure Matters

In many graphs, especially citation and social networks, nodes that are connected tend to share similar labels. This property is known as **homophily**. The Cora dataset exhibits strong homophily, as papers on similar topics frequently cite one another.

Ignoring graph structure removes this valuable signal. By incorporating neighborhood information, GNNs exploit homophily to improve predictive performance.

4 The Cora Dataset

Cora is a widely used benchmark dataset for node classification tasks. It consists of:

- Nodes representing scientific publications
- Edges representing citation relationships
- Node features given as bag-of-words vectors
- Seven distinct research topic labels

The objective is to predict the research topic of each paper using both its textual content and its citation links.

5 Graph Learning Framework

Most graph learning tasks can be expressed using four components: an encoder, a decoder, a ground-truth function, and a loss function.

5.1 Encoder (Enc)

The encoder maps the input graph and node features to latent embeddings:

$$\text{Enc} : (G, X) \rightarrow Z = \{z_i\}$$

where z_i represents the embedding of node i. In GNNs, the encoder performs message passing across edges.

5.2 Decoder (Dec)

The decoder maps embeddings to task-specific outputs, such as class probabilities for node classification.

5.3 Ground-Truth Function (G_t)

The ground-truth function provides target labels for nodes, edges, or entire graphs.

5.4 Loss Function (L)

The loss function compares predictions with ground truth and is minimized during training:

$$L(G_t, \text{Dec}(Z))$$

6 Models

6.1 MLP Baseline

The MLP baseline ignores graph structure and processes each node independently.

- Enc: Feed-forward neural network
- G_t: Identity (graph ignored)
- Dec: Linear classifier
- L: Cross-entropy loss

This baseline quantifies the benefit of incorporating relational information.

6.2 Graph Convolutional Network (GCN)

GCN aggregates information from a node and its neighbors using a normalized adjacency matrix.

- Enc: Linear transformation with normalized neighborhood aggregation
- G_t: Degree-normalized adjacency matrix
- Dec: Linear output layer
- L: Cross-entropy loss

GCN assumes all neighbors contribute equally and performs well on homophilic graphs.

6.3 GraphSAGE

GraphSAGE aggregates neighbor information using a learnable aggregation function.

- Enc: Concatenation of node and aggregated neighbor embeddings
- G_t: Learnable aggregation (e.g., mean)
- Dec: Linear classifier
- L: Cross-entropy loss

GraphSAGE supports **inductive learning**, allowing generalization to unseen nodes or graphs.

6.4 Graph Attention Network v2 (GATv2)

GATv2 introduces attention to learn the importance of neighbors dynamically.

- Enc: Attention-weighted neighbor aggregation
- G_t: Learned attention coefficients
- Dec: Linear classifier
- L: Cross-entropy loss

Attention allows the model to focus on informative neighbors and reduce noise.

7 Experimental Setup

7.1 Data Splits

The dataset includes predefined training, validation, and test masks. Models are trained on training nodes, tuned using validation nodes, and evaluated on test nodes.

7.2 Hyperparameters

- Optimizer: Adam
- Learning rate: 0.01 (GCN, GraphSAGE), lower for GATv2
- Hidden dimension: 64 (GCN, GraphSAGE)
- GATv2: 8 attention heads
- Dropout: 0.5 (GCN, GraphSAGE), 0.6 (GATv2)
- Epochs: 200–300

8 Results

Model	Test Accuracy
GCN	0.8050
GraphSAGE	0.7730
GATv2	**0.8280**

Table 1: Node classification accuracy on the Cora dataset.

9 Discussion

The MLP baseline performs significantly worse, highlighting the importance of graph structure. GCN achieves strong performance due to its simplicity and effectiveness on homophilic graphs. GraphSAGE performs slightly worse, as its inductive capabilities are less critical in this transductive setting.

GATv2 achieves the highest accuracy by learning attention weights that prioritize informative neighbors. However, this increased expressiveness comes at the cost of higher computational complexity and sensitivity to hyperparameters.

10 Limitations and Over-Smoothing

A known limitation of GNNs is **over-smoothing**, where node embeddings become indistinguishable as more layers are added. This limits the depth of practical GNN architectures and motivates research into residual connections and normalization techniques.

11 Conclusion

This study demonstrates that incorporating graph structure significantly improves node classification performance. Among the evaluated models, GATv2 performs best due to its attention-based aggregation, while GCN remains a strong and efficient baseline. The results emphasize that model choice depends on dataset properties, graph homophily, and computational constraints.