Reproducibility of GAT Experiments

1 Introduction

Graph-structured data are prevalent in various domains such as social networks, citation networks, biological networks, and knowledge graphs. Traditional neural networks are not directly applicable to such data due to their irregular structure. To address this, Graph Neural Networks (GNNs) have been developed to extend deep learning techniques to graph data.

The Graph Attention Network (GAT) proposed by Veličković et al. (2018) introduces an attention mechanism to GNNs, allowing nodes to attend to their neighbors' features with different weights. This method leverages masked self-attention layers to address the shortcomings of prior spectral-based approaches like Graph Convolutional Networks (GCNs).

Key contributions of the GAT paper include:

- **Attention Mechanism on Graphs:** Introducing a novel attention-based layer that allows for assigning different importances to different nodes in a neighborhood, enhancing the expressive capability of GNNs.
- **Efficiency and Parallelization:** The GAT model is computationally efficient and parallelizable, as it does not require costly matrix operations like eigendecomposition.
- **Inductive and Transductive Learning:** Demonstrating that GATs can be applied to both inductive tasks (where the model generalizes to unseen graphs) and transductive tasks (where the entire graph is known during training).
- **State-of-the-Art Performance:** Achieving or matching state-of-the-art results on benchmark datasets such as Cora, Citeseer, Pubmed (transductive learning), and a protein-protein interaction dataset (inductive learning).

In this report, we replicate the experiments conducted in the GAT paper for transductive learning tasks on citation networks. Specifically, we focus on the Cora, Citeseer, and Pubmed datasets. Our goal is to verify the reproducibility of the reported results and assess the effectiveness of the GAT model in classifying nodes in citation networks.

2 Scope of Reproducibility

This section defines the hypotheses from the original paper that we aim to validate and outlines the experiments we conducted to test these claims.

2.1 Hypotheses

The original GAT paper posits the following key claims:

1. **Performance on Transductive Learning Tasks:** GATs achieve or exceed state-of-the-art performance on the Cora, Citeseer, and Pubmed datasets in node classification tasks. Reported classification accuracies are:

$$\begin{aligned} \text{Cora: } & 83.0\% \pm 0.7\% \\ \text{Citeseer: } & 72.5\% \pm 0.7\% \\ \text{Pubmed: } & 79.0\% \pm 0.3\% \end{aligned}$$

2. **Effectiveness of Multi-Head Attention:** Multi-head attention stabilizes the learning process and enhances model performance by aggregating diverse neighborhood representations.

3. **Expressive Capacity of Attention Mechanism:** The attention mechanism enables the model to assign different importances to neighbors, improving classification accuracy compared to non-attentional approaches like GCNs.

2.2 Experiments

To test these hypotheses, we replicate the following experiments from the original paper:

1. **Dataset Selection:** Transductive learning tasks on the Cora, Citeseer, and Pubmed citation networks, where nodes represent documents, edges represent citations, and node features are bag-of-words representations.

2. **Model Training:**

   - Implementation of a two-layer GAT model with multi-head attention in the hidden layers.
   - Hyperparameters:
     - Learning rate: 0.005
     - Dropout: 0.6
     - Weight decay: 5×10^{-4}
     - Attention heads: 8 in the hidden layer, 1 in the output layer
     - Hidden layer dimension: 8 features per attention head

3. **Evaluation Metrics:** Classification accuracy on the test set for each dataset. Comparison of results with the baseline GAT performance reported in the paper.

4. **Reproducibility Challenges:** Analysis of factors affecting reproducibility, including data preprocessing, hyperparameter tuning, and computational resources.

3 Methodology

3.1 Model Description

The GAT model enhances node representation learning by aggregating information from neighboring nodes using a dynamic attention mechanism.

Architecture:

- **First Layer:** 8 attention heads, each producing an 8-dimensional output per node. Outputs are concatenated to form a 64-dimensional vector per node.
- **Second Layer:** A single attention head aggregates features to produce final logits, normalized with a softmax function for class prediction.

Attention Mechanism:

1. Input Transformation: Node features $\mathbf{h}_i$ are transformed using a learnable weight matrix $\mathbf{W}$.
2. Concatenation: Transformed features of nodes i and j are concatenated:

$$\mathbf{e}_{ij} = [\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_j]$$

3. Shared Attention Mechanism:

$$e_{ij} = \text{LeakyReLU}\left(\mathbf{a}^\top \mathbf{e}_{ij}\right)$$

4. Coefficients Normalization:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$

5. Feature Aggregation:

$$\mathbf{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\mathbf{h}_j\right)$$

where σ is a non-linear activation function (ELU in hidden layers).

3.2 Dataset Description

The experiments used the Cora, Citeseer, and Pubmed citation network datasets:

- **Cora:** 2,708 nodes, 5,429 edges, 1,433 features, 7 classes.
- **Citeseer:** 3,327 nodes, 4,732 edges, 3,703 features, 6 classes.
- **Pubmed:** 19,717 nodes, 44,338 edges, 500 features, 3 classes.

3.3 Data Preprocessing

- Row normalization of feature matrices.
- Symmetric normalization of adjacency matrices:

$$\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

where A is the adjacency matrix, and D is the degree matrix.

4 Computational Implementation

Framework:

- Python with PyTorch for model building.
- NumPy and SciPy for numerical and sparse matrix operations.

Hardware: MacBook Pro M3 Pro with 16GB unified memory.
Hyperparameters:

- Learning Rate: 0.005
- Dropout: 0.6
- Weight Decay: 5×10^{-4}
- Attention Heads: 8 (hidden layer), 1 (output layer)
- Activation Function: LeakyReLU ($\alpha = 0.2$)
- Epochs: 1,000 with early stopping (patience = 100 epochs)