Deep Learning – Lab 06

**1. NetworkX Random Graph Analysis**

**Graph Density Calculation:**

* I increased the number of nodes (N) from **20** to **200** and computed the graph density for multiple values of N. The graph density, which measures how many edges are in the graph compared to the maximum possible number of edges, decreased as the number of nodes increased.

**Degree Distribution Observation:**

* I plotted the degree distribution for different values of N. As N increased, the degree distribution histogram showed more variation, meaning that the degree distribution became more spread out as the network grew in size.

**2. Learning Method Comparisons**

**Supervised vs. Self-supervised vs. Semi-supervised Learning:**

* **Supervised learning** requires fully labeled data.
* **Self-supervised learning** generates its own labels from the data for pretext tasks.
* **Semi-supervised learning** uses a combination of labeled and unlabeled data, as in the Karate Club GCN task where only part of the data is labeled.

**Transductive vs. Inductive Learning:**

* **Transductive learning** uses the entire graph (including test nodes) during training, making it specific to the test set (used in the Karate Club dataset).
* **Inductive learning**, on the other hand, generalizes to unseen data.

**3. Karate Club GCN (Graph Convolutional Networks)**

**Experiment 1 - Increasing Epochs:**

* I increased the number of epochs from **50 to 500** in the training process.
* **Observation**: The validation accuracy improved slightly over time. However, the improvement became negligible after about 300 epochs, indicating that 500 epochs might lead to diminishing returns in accuracy.

**Experiment 2 - Self-loops in GCNConv:**

* I experimented by **removing self-loops** from the GCNConv() layers.
* **Observation**: The model accuracy **decreased** slightly without self-loops, indicating that self-loops may have contributed to better performance.

**Experiment 3 - Increasing GCNConv Layers:**

* I extended the GCN model from **3 layers to 8 layers**.
* **Observation**: Initially, increasing the layers improved the accuracy, but after 5 layers, the performance started to degrade. This suggests that adding more layers without proper regularization may lead to **overfitting** or **vanishing gradients**.

**Experiment 4 - Hyperparameter Tuning:**

* I adjusted the in\_channels and out\_channels in GCNConv() and found that smaller values for out\_channels led to faster training but slightly lower accuracy. I selected **16** as an optimal value for both in\_channels and out\_channels.

**Experiment 5 - Adding Skip Connections:**

* I added skip connections between some of the GCNConv() layers to improve the model’s performance.
* **Observation**: Adding skip connections stabilized the learning process and improved validation accuracy. It helped in mitigating the degradation that occurred with deeper layers.

**4. Comparison of GNN Models**

**Message Passing GNN, GCN, GAT, and GraphSAGE:**

* **Message Passing GNN**: Relies on iteratively passing node features to neighbors.
* **GCN (Graph Convolutional Network)**: Utilizes convolutional layers that aggregate node information based on neighbors.
* **GAT (Graph Attention Network)**: Introduces attention mechanisms to weigh neighboring node contributions.
* **GraphSAGE**: Samples a fixed number of neighbors and aggregates them, which makes it scalable to large graphs.