**DL lab 6 – Graph Neural Networks**

**1.** **Change in Validation Accuracy with Increased Epochs**

**Experiment:** Increased the number of training epochs from 50 to 500 to observe how prolonged training impacts the model's performance.

**Observation:** Initially, as the number of epochs increased from 50 to around 200, the validation accuracy showed steady improvements.No significant improvement was observed beyond 400 epochs, and at times, slight overfitting occurred.

**2. Model Accuracy Without Self-Loops in GCNConv Layers**

**Experiment:** Removed the self-loops from the GCNConv() layers (by setting add\_self\_loops=False) to determine the impact on model performance.

**Observation:** Without the self-loops, the validation accuracy dropped .This decrease suggests that self-loops play a crucial role in allowing nodes to aggregate their own features during message passing. The absence of self-loops negatively impacted the model's ability to learn important node-specific features, leading to lower accuracy.

**3. Impact of Increasing GCNConv Layers from 3 to 8**

**Experiment:** We increased the number of GCNConv() layers in the model from the original 3 to 8 to see how deeper architectures perform on the dataset.

**Observation:** increasing the number of layers to 5, the accuracy rose slightly.However, as the layers were increased to 8, the accuracy dropped back .

**4. Effect of Tuning In\_channels and Out\_channels in GCNConv Layers**

**Experiment:** We experimented with different values for the in\_channels and out\_channels hyper-parameters in the GCNConv() layers.

**Observation:** Further increasing the channels (e.g., 64 and 128) led to slightly worse results, possibly due to overfitting or the model becoming too complex for the dataset. Finding the optimal channel sizes is key for maximizing performance.

**5. Effect of Adding Skip Connections Between GCNConv Layers**

**Experiment**: We introduced skip connections between certain GCNConv() layers to see if bypassing some layers could improve the model's performance.

**Observation:** The skip connections helped mitigate the vanishing gradient problem by allowing gradients to flow more easily through the network, improving performance. The model benefited from the skip connections as it allowed the model to use both shallow and deep features effectively.

1. Explain the differences between Message Passing GNN, graph convolution network (GCN), graph attention network (GAT) and GraphSAGE. Write the answers in the word file.
2. **Message Passing Neural Networks (MPNN):**

**Overview:** MPNNs are a general framework for graph neural networks where information is passed between nodes. The message-passing process is a two-step procedure consisting of message aggregation and update steps.

**Key Feature:** It is highly flexible, allowing various types of aggregation and update functions, making it a generalization of many GNN models.

**Use Case:** It’s well-suited for tasks where customized aggregation methods are required.

1. **Graph Convolutional Network (GCN):**

**Overview:** GCN is one of the foundational models in GNNs that generalizes the convolution operation from Euclidean space to graph structures. It aggregates feature information from neighboring nodes using spectral methods.

**Key Feature:** Uses normalized adjacency matrices to propagate node information and capture local neighborhood structures.

**Use Case:** Useful in semi-supervised classification, node classification, and link prediction tasks.

1. **Graph Attention Network (GAT):**

**Overview**: GAT extends GCN by incorporating attention mechanisms, allowing the network to weigh the importance of neighboring nodes when aggregating information.

**Key Feature:** Introduces attention weights to improve the focus on relevant nodes during the aggregation process.

**Use Case:** Works well for problems where neighboring nodes have different levels of importance, such as social networks or recommendation systems.

1. **GraphSAGE:**

**Overview:** GraphSAGE is designed to sample and aggregate information from a fixed-size local neighborhood of each node, making it scalable to large graphs.

**Key Feature**: It uses inductive learning, allowing it to generate embeddings for previously unseen nodes by learning an aggregation function.

**Use Case**: Effective for large-scale graph learning and generalization to new nodes.