# LAB 6 – ANSWERS – IT21059940

Github repo link - <https://github.com/it21059940/DL_Lab6_IT21059940>

## Question 1

Explanation

As the number of nodes (N) increases (eg from 20 to 200), the graph density goes down. For example, with 20 nodes, the density is 0.21, but with 200 nodes, it’s much lower at 0.020. This is because while the number of connections grows, the possible connections increase much faster, making the graph less dense.

In the degree distribution (how many connections each node has), when N=20, the degrees vary more, with some nodes having few connections and others having many. But with N=200, the degrees are more balanced, with most nodes having about the same number of connections

Question 2

Part 1

1. Supervised Learning - In supervised learning, the model is trained on labeled data, each input data has a label. The model learns to predict the correct label for new, unseen data based on this training.

Eg – Classification into cats and dogs, after being trained with labelled images of cats and dogs.

1. Self-Supervised Learning - In self-supervised learning, the model trains itself using unlabeled data. It creates its own labels by setting up tasks like predicting missing parts of the data or generating outputs from partial inputs

Eg - predicting the next word in a sentence.

1. Semi-Supervised Learning - Semi-supervised learning is a combination of supervised and unsupervised learning. The model is trained using a small amount of labeled data and a large amount of unlabeled data. The labeled data helps guide the learning process, while the unlabeled data helps the model generalize better and make more accurate predictions.

Part 2

In inductive learning, the model learns a general pattern from the training data and applies that rule to make predictions on unseen data. The goal is to learn a function that can work on any future data point. For example, in supervised learning, a model trained on labeled images of cats and dogs will try to generalize and correctly classify new, unlabeled images.

In transductive learning, the model is specifically concerned with predicting labels for a given set of known test examples, rather than learning a general rule. It doesn't generalize to new, unseen examples beyond the provided test set. The idea is to make predictions only for the data points provided, like predicting the labels for a fixed set of test examples using both training and test data.

So in conclusion, inductive learning aims to generalize beyond the current data set, while transductive learning focuses only on making predictions for a known test set without wanting to generalize further.

## Question 3

When the number of epochs is increased from 50 to 500, the validation accuracy increases. Initially at epoch=40 validation accuracy was 64.71% as shown below.

A black text on a white background

Description automatically generated

When epochs was increased to 500 validation accuracy increased to 82.35% as shown below.

A close-up of a message

Description automatically generated

Use of self-loops

Self-loops are edges that connect a node to itself. In Graph Convolutional Networks (GCNs), self-loops are often added to ensure that a node's feature representation includes its own features during the convolution operation

Removing self-loops

from torch\_geometric.utils import remove\_self\_loops

#ADDITIONAL CODE TO REMOVE SELF-LOOPS

data.edge\_index, \_ = remove\_self\_loops(data.edge\_index)

By default, self-loops are enabled, by disabling self-loops the accuracy slightly drops.

Increasing the number of layers

When the number of layers was increased to 8 the model became too complex and therefore resulted in a drop in accuracies. Adding skip connections is like regularization, it helped the model generalize a bit better and therefore improved the accuracies.

## Question 4

Message Passing GNN (MP-GNN)

Message Passing GNNs work by nodes exchanging information with their neighbors. Each node updates its state by aggregating information from its neighbors, passing messages up and down, repeating this process over several rounds. This captures relationships between nodes.

Graph Convolution Network (GCN)

GCNs are like regular Convolutional Neural Networks (CNNs) but for graph data. Each node updates its state by taking a weighted average of its neighbors' features. This helps understand the local structure of the graph more easily.

Graph Attention Network (GAT)

GATs improve GCNs by assigning different importance (attention) to neighbors. GATs learn to focus more on important neighbors, which helps capture more useful information for each node by paying more attention to relevant ones.

GraphSAGE

GraphSAGE is designed for large graphs by sampling a fixed number of neighbors for each node, instead of considering all neighbors. It generates node embeddings by sampling and aggregating neighbors, making it better for large-scale graph data.