**GitHub repository link: -** [**https://github.com/GayashanDeshapriya/DL-Lab06-IT21156656.git**](https://github.com/GayashanDeshapriya/DL-Lab06-IT21156656.git)

1. **. Increase the N value from 20 (original value) to 200 with multiple N values in between and observe the change of graph density and degree distribution (i.e., histogram plot). Explain what you observe and write the answer in a word file.**

When increase the N value (number of nodes) from 20 to 200 while keeping the ratio of edges to nodes constant (E = 2\*N), we can observe the following things: -

Graph Density: The graph density will decrease as N increases. This is because the number of possible edges in a graph increases much faster than the number of actual edges as you add more nodes. The formula for density is D = 2 \* |E| / (|V| \* (|V| - 1)), and since you're keeping E proportional to N, the denominator grows much faster than the numerator.

Degree Distribution: The degree distribution will become more concentrated around the average degree. With more nodes and a constant edge-to-node ratio, the degrees of individual nodes are likely to be closer to the average. The degree distribution plots (histograms) will likely show a taller peak around the average degree and a narrower spread as N increases.

In simpler terms, as we add more nodes to the graph while maintaining the same ratio of edges to nodes, the graph becomes more sparse (less dense). The connections become more evenly distributed, and most nodes have a similar number of connections (degree).

1. **.**
2. **. Explain the differences between supervised learning, self-supervised learning and semi-supervised learning methods**

**Supervised Learning**

* Supervised learning uses a labeled dataset to train an algorithm to map an input to an output. For example, if you wanted to classify images of cats and dogs, you would need a dataset of images labeled as "cat" or "dog".

**Self-supervised learning**

* Self-supervised learning is a type of unsupervised learning where the algorithm learns from the data itself, without any explicit labels. For example, an algorithm could be trained to predict the next word in a sentence, given the previous words.

**Semi-supervised learning**

* Semi-supervised learning uses a combination of labeled and unlabeled data to train an algorithm. This is often used when there is a limited amount of labeled data available. For example, we could use a small dataset of labeled images to train an algorithm to classify images, and then use a larger dataset of unlabeled images to improve the algorithm's performance.

1. **. Explain the differences between transudative learning and inductive learning.**

**Transudative learning**

* Transudative learning aims to make predictions only on the specific data points that are available during training. The model does not learn a general rule but focuses on making the best predictions for the given data.

**Inductive learning**

* Its aims to learn a general rule from the training data that can be applied to any new data points. The model tries to generalize from the training data to learn a function that can be used to make predictions on unseen data.

**Increase the number of epochs from 50 to 500 and observe the change in validation accuracy and write what you observe in the word file.**

More epochs could give a further increase in the accuracy of this model on the validation set. The case is that iterating for more epochs may lead to starting higher but then eventually decreasing or fluctuating validation accuracy. In the case of further iteration over more epochs, the model's validation accuracy could get 'saturated', which means after a certain number of epochs, it stops improving significantly.

**Experiment without self-loops added to GCNConv() layers in the GCN() model and detail the model accuracy increase/decrease**

In these cases, removing self-loops lead to a decrease in model accuracy. Self-loops help nodes retain their own information during message passing, which can be important for tasks where node features are highly informative.

With self-loop

* training accuracy =100.00 % (for 500 epochs)
* validation accuracy =82.35 % (for 500 epochs)

Without self-loop

* training accuracy =100.00 % (for 500 epochs)
* validation accuracy =61.76 % (for 500 epochs)

1. **.**
2. **Increase the number of GCNConv() layers in the GCN() model upto 8 layers from original 3 layers. Detail the accuracy increase/decrease in the word file.**

When increasing the number of GCNConv() layers in the GCN() model up to 8 layers from original 3 layers the accuracy increases while validation accuracy decreases.

* + - validation accuracy =55.88 % (for 500 epochs)

1. **. Add skip connections between some of the GCNConv() layers and try to see if that can improve the model performance.**

It seems that for configurations of the size of the channels, the skip connections do lead to even better validation accuracy compared to the base. Adding the skip connections between some of the GCNConv() makes a difference in getting even better performance for the model.

* validation accuracy =79.41 % (for 500 epochs)

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.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Feature/Aspect** | **Message Passing Neural Network (MPNN)** | **Graph Convolutional Network (GCN)** | **Graph Attention Network (GAT)** | |  | | --- | |  |  |  | | --- | | **GraphSAGE** | |
| **Key Idea** | General framework for learning node representations by iteratively exchanging messages between neighbors. | Extends classical convolution to graphs by aggregating node features from neighbors. | Uses attention mechanisms to weigh neighbor contributions in the aggregation process. | Learns node embeddings by sampling and aggregating features from a fixed-size neighborhood. |
| **Message Passing Scheme** | |  | | --- | | Explicitly defines a message and update function between nodes to propagate information. | | |  | | --- | | Simplified as averaging or weighted sums of neighbors. | | |  | | --- | | Attention scores are computed for each neighbor to determine their contribution. |  |  | | --- | |  | | Uses neighbor sampling and concatenation/aggregation of features. |

* MPNN is a general framework that can be customized for various tasks but may require more complex implementation.
* GCN is simple and efficient, using a weighted average of neighbors but lacks attention to different node importance.
* GAT incorporates attention mechanisms, making it more expressive but also more computationally expensive.
* GraphSAGE introduces a scalable approach by sampling neighbors, making it suitable for inductive learning and large graphs.