DL lab 6 – Graph Neural Networks

1. Upload the NetworkX jupyter notebook file (i.e.,NetworkX\_tutorial.ipynb) to google colab root directory.
   * Run the above code and understand it.
   * Complete the code sections to get the degree matrix and Laplacian matrix of the created random graph.
   * Calculate the graph density of the random graph in the code. Use the below equation (D = graph density, |V| = number of nodes and |E| = number of edges).
   * 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.



Answer : Graph density has been descreased

: Degree distribution has became more valid

1. In the KarateClub dataset based GCN code, we use semi-supervised training approach along with the transductive leaning method.

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

**Supervised Learning**:

* Uses labeled data to train a model to predict outputs for given inputs
* Requires a large amount of manually annotated training data, which can be costly and time-consuming
* Performs well on specific tasks with sufficient labeled data

**Self-Supervised Learning**:

* Uses unlabeled data to automatically generate labels or "pseudo-labels" to train a model
* Learns meaningful representations from the underlying structure of the unlabeled data
* Allows models to be pre-trained on large amounts of unlabeled data and then fine-tuned on smaller labeled datasets
* Techniques include self-predictive learning (predicting missing parts of data) and contrastive learning (comparing similar and dissimilar data)

**Semi-Supervised Learning**:

* Uses a combination of labeled and unlabeled data to train a model
* Starts with a small amount of labeled data to train an initial model, then iteratively applies the model to unlabeled data to generate pseudo-labels
* Adds the most confident pseudo-labels to the labeled dataset to train an improved model
* Reduces the need for manual annotation compared to supervised learning
* Can be used for a variety of tasks like classification, regression, clustering, and association
  + Explain the differences between transductive learning and inductive learning.

**Transductive Learning**:

* Aims to predict the labels of a fixed set of unlabeled examples
* Makes predictions only for the given unlabeled examples, not for new unseen examples
* Utilizes the structure of the unlabeled data to improve predictions
* Computationally efficient as it only needs to make predictions for the given unlabeled set

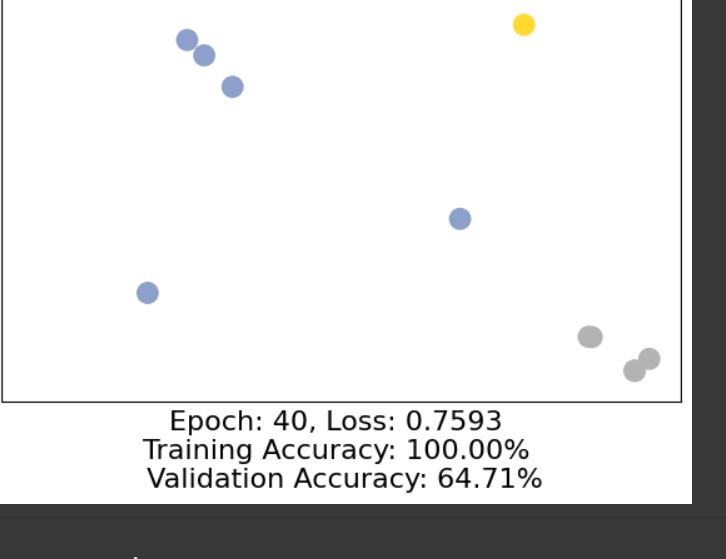
**Inductive Learning**:

* Learns a general function that can make predictions for any new unseen examples
* Builds a model that can generalize beyond the training data
* Requires labeled data to learn the underlying patterns and relationships
* Computationally more expensive as it needs to learn a general function

Upload the KarateClub dataset based GCN jupyter notebook file (i.e., KarateClub\_GCN\_introduction.ipynb ) to google colab root directory.

* + In this code, we use Zachary’s karate club network dataset.
  + Run the above code and understand it.
  + 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.

With 50 epochs



With 500 epochs

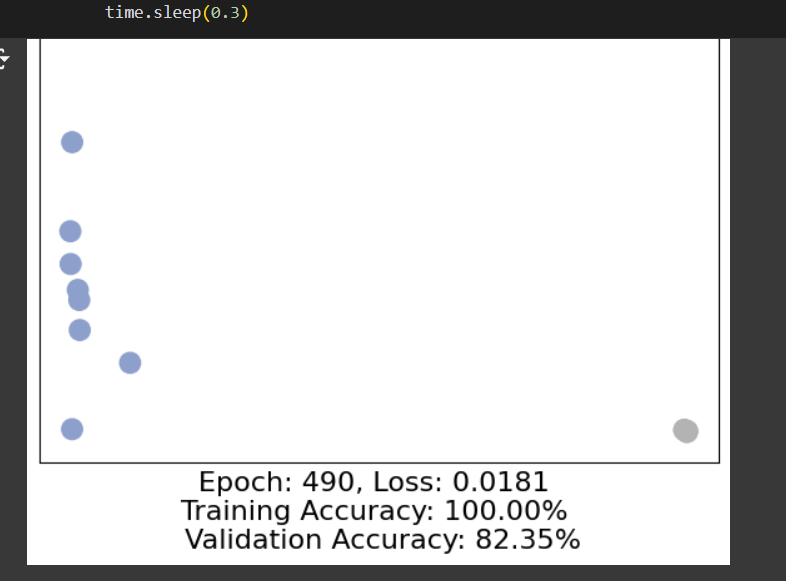
A screenshot of a computer

Description automatically generated

With 500 epochs the validation accuracy is considerably higher than 50 epochs . training accuracy is similar but with 500 epochs loss is considerably lower than 50 epochs

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

**With self loops**



**Without self loops**

A screenshot of a computer

Description automatically generated

Removing self-loops from the GCNConv layers in the GCN model decrease in model accuracy.

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.

Accuracy Observations

1. **Initial Model (3 Layers)**:

A screenshot of a computer

Description automatically generated

1. **Modified Model (8 Layers with Skip Connections)**:

A screenshot of a computer

Description automatically generated

* + 1. In\_channels and out\_channels in GCNConv() can be considered as hyper-parameters and you can use the best performing values you find.
    2. Add skip connections between some of the GCNConv() layers and try to see if that can improve the model performance.
    3. Detail what you observe in the word file.

Conclusion

* The addition of layers and the implementation of skip connections improved the model's performance.
* This indicates that deeper architectures with skip connections can help in better feature learning and mitigate issues like vanishing gradients, allowing the model to capture more complex relationships in the data.

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.

1. Message Passing Graph Neural Networks (MPNN)

* **Concept**: MPNN is a general framework for graph neural networks that focuses on the idea of passing messages between nodes in a graph. Each node aggregates messages from its neighbors to update its representation.
* **Mechanism**: Nodes send and receive messages based on their features and the features of their neighbors. The aggregation function can vary and can be learned or predefined.
* **Flexibility**: MPNNs can be tailored for various tasks and can incorporate different types of message aggregation and update functions.

2. Graph Convolutional Network (GCN)

* **Concept**: GCN is a specific type of MPNN that uses convolutional operations to aggregate node features.
* **Mechanism**: It employs a normalized sum of the features of neighboring nodes, including the node itself, to produce the new representation. The aggregation is based on the graph structure and is defined by the graph Laplacian.
* **Limitations**: GCNs can suffer from over-smoothing, where deeper layers lead to indistinguishable node representations, and they are less flexible in handling varying neighborhood structures.

3. Graph Attention Network (GAT)

* **Concept**: GAT introduces an attention mechanism to the message passing framework, allowing the model to weigh the importance of different neighbors when aggregating their features.
* **Mechanism**: Each node computes attention scores for its neighbors, which are then used to perform a weighted aggregation of their features. This allows the model to focus on more relevant neighbors.
* **Advantages**: GATs can handle graphs with varying neighborhood structures better than GCNs and can learn to assign different importance to different neighbors, improving generalization.

4. GraphSAGE

* **Concept**: GraphSAGE (Graph Sample and Aggregation) is designed for inductive learning on large graphs. It samples a fixed-size neighborhood for each node to perform aggregation.
* **Mechanism**: It allows for different aggregation functions (mean, LSTM, pooling) and learns to combine features from sampled neighbors, which makes it scalable to large graphs.
* **Flexibility**: GraphSAGE can generalize to unseen nodes and graphs, making it suitable for dynamic graph scenarios where the structure may change over time.

**Submission.**

Download the final modified notebook files (all 2 jupyter notebooks). Add these notebooks and the word file to a new zip file. Upload this zip file to the courseweb submission link. The file name should be your registration number.