**2. Motivation:**

GNNs have been shown to be effective in the prediction of complex behaviors of a system. Despite their abilities, they are still new models in the machine learning field and there are many open questions related to their performance, interpretability, and training difficulties.

In this paper, we analyze existing popular GNNs for different types of graph datasets and then investigate these models from different aspects such as training convergence, the effect of the convolutional layer, initialization methods, activation functions, and dropout rate.

**3. Methodology:**

In this section, we present the way we have conducted our experiments. The main task we have considered conducting experiments based on is Node Classification. Node-level tasks have the goal to classify nodes in a graph. Usually, we have given a single, large graph with >1000 nodes of which a certain number of nodes are labeled. We learn to classify those labeled examples during training and try to generalize to the unlabeled nodes.

We considered three popular datasets in graph node-level prediction which all belong to the relationship among paper citations. Further details related to the number of nodes and edge is summarized in table 1:

Table 1: Datasets and their sizes

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Cora | CiteSeer | PubMed |
| Node Size | 2700 | 3327 | 19717 |
| Edge Size | 1433 | 3703 | 500 |

As an example, relationships for the Cora dataset are illustrated in Fig 1.

Diagram

Description automatically generated

Figure 1: Graph Relationships in the Cora Dataset

Based on the purpose of this paper, we analyze training difficulties from various aspects. First, we consider four types of convolutional layers for graphs: Graph Convolution Networks, Simplified Graph Convolutions, Multi-head Graph Attention Networks (8 heads), and Residual Gated Graph Convolutional.

Being specific about difficulties related to training GNNs, we considered analyzing the effects of Activation Functions, Initialization Methods, and Dropout Rates on training GNNs. For this purpose, we develop specific classes based on PyTorch geometric framework with two layers. The first is one of the four convolution layers with 5 hidden nodes and the second is a linear fully connected layer.

Additionally, we analyzed the effect of getting deep into convolutional layers to observe the output range for each hidden graph convolution layer. For this purpose, we add five GCN convolution layers and one as a hidden layer with 50 node size, and one linear fully connected layer.

For the activation function, we consider four popular activation functions: Sigmoid, Tanh, Rectified Linear Units (Relu), and Leaky Relu with a slope coefficient of 0.1. For initialization methods, our considered options are Random initialization (default), Uniform with a range of [0,1], Glorot (Xavier), and Kaiming Uniform. Finally, for dropout analysis, we considered five values including 0.1, 0.3, 0.5, 0.7, and 0.9 which apply before each layer.

It is worth mentioning that the selected optimizer is Adam with a learning rate of 0.1. We also add a weight penalty of 0.005 as an L2 penalty to prevent overfitting. The loss function is also Cross Entropy. For training difficulties, the number of epochs is set to 1,000, but for deep hidden convolution layers analysis, the number of epochs is 10,000.

Overall, for each of these sections, a node classification task will be run and then the results will be compared in terms of accuracy.

**4. Experiments and Results:**

In this section, the results of the training difficulties of GNNs are presented. The detailed results for each experiment can be found on the GitHub repository[[1]](#footnote-1)

**4.1. Results on Initialization Methods**

As a first analysis, the effect of applying different initialization methods on the linear layer of the graph convolutional model is investigated. An important point is that due to the convolution structure of almost all GNNs models which is a matrix multiplication of adjacency matrix and some data-driven matrices like Laplacian matrix, defining the initialization method directly for convolution layers is tricky. As a result, we apply initialization to the output of convolution layers. Figure 2 shows re results for three datasets and four different layers.

Graphical user interface, application

Description automatically generated

Figure 2. Initialization Method Comparison for three datasets and four layers

An important point based on the observation of figure 2 is that, as the complexity of the data increase, the effect of the type of initialization methods tend to vanish in the final output and accuracy.

**4.2. Results on Activation Functions**

For this part of the analysis, different activation functions are applied to the output of each layer (Conv & Linear). The results are illustrated in Figure 3.

Graphical user interface, application

Description automatically generated

Figure 3. Activation Function Comparison for three datasets and four layers

The key point here is that, in simple datasets like Cora, there is no big difference among different activation functions, however, as the dataset gets more complicated, Relu and leaky Relu start to show better results.

**4.3. Results on Dropout Rate**

The effect of increasing dropout rate can show us important hints about the sparse qualities of graphs and the importance of connections among different nodes in the graph data. Figure 4 demonstrate accuracy results for different datasets and layers as the dropout rate increase.

Graphical user interface

Description automatically generated

Figure 4. Dropout Rate Comparison for three datasets and four layers

As expected, accuracy plummeted as the dropout rate increased, however, the key finding is that, although dropout only applies to hidden layers, but the ratio of the node to edge is quite important in the model’s robustness when a dropout rate increases. Moreover, GAT shows more robustness for different dropout rates.

**4.4. Results on Deep Convolution Layers**

As a different analysis from previous ones, we investigate the effect of getting deeper in terms of increasing the number of layers in GNNs. Every 100 epochs, the output of five GCN model layers is extracted and aggregated. Figure 5 shows the average results for three datasets.

Graphical user interface, application, Teams

Description automatically generated

Figure 5. Effect of going deep in GCN

Based on figure 5, going deep into more than three convolution layers leads the output to take extremely large values after the 2000th epochs. First, we hypnotize that this phenomenon is a result of overfitting but after analyzing the accuracy results, we found that after getting deeper, accuracy increased by around 3%. As an alternative, This phenomenon can lead us to conclude that deeper layers get more important in terms of accuracy. This phenomenon should be analyzed further.

**Conclusion**

d

Changing the optimization method from Adam to SGD didn't make any difference

Generally, GAT shows faster convergence to stable accuracy and more resilience in dropout

1. https://github.com/hadiagha/GNNProject [↑](#footnote-ref-1)