# Understanding the Difficulty of Training Graph Neural Networks

Hadi Aghazadeh, Amirhossein Ahmadi, Fatma Mutlu Kipirti

December 7, 2022

## Abstract

The data in classical deep learning tasks are typically represented in Euclidean space. However, there are many data with non-Euclidean domains which can be represented as graphs with complex relationships and interdependency between objects. Moreover, training machine learning methods with graph-structured data is a great challenge. Graph embedding algorithms are a remedy for making graph data more Euclidean but they do not have the inductive ability to extract semantic information for prediction purposes. Recently, Graph Neural Networks (GNNs) have demonstrated ground-breaking performances. Like Convolution Neural Networks (CNNs), these models take graphs in their non-Euclidean form and embed them in Euclidean space in a learnable process that enables them to be merged with the power of classical deep learning models. In this paper, we investigated training difficulties for GNNs. Results show that those training configurations that address locality in GNN models have more contribution for getting higher accuracy in graph node-level prediction tasks.

## 1 Introduction

Datasets in deep learning applications such as video processing (video), image processing (image), speech recognition (speech), natural language processing (text), and forecasting (time series) are typically sequences represented in the Euclidean space with the same structure and size. However, as can be seen in Fig. 1, there is an increasing number of applications where data are generated from non-Euclidean domains and are represented as graphs with complex relationships (i.e., edges) and interdependency between objects (i.e., nodes). Graphs are powerful data structures that allow us to easily express these relationships without a fixed form between unordered nodes with a variable size. They can be categorized into directed vs undirected, weighted vs unweighted, and simple vs multigraphs. Using this data in a conventional deep learning model has proven to be not easy to directly make use of the structural information of graphs in these applications as graph data are high-dimensional and non-Euclidean.

GNN allows the creation of an end-to-end machine learning model that is a class of deep learning for processing data represented by graphs. It combines node feature information with the graph structure by recursively passing neural messages along the edges of the input graph. The idea behind GNNs aims to learn graph neural networks embeds nodes for a low-dimensional and non-Euclidean data so that geometric relationships in the embedding space reflects the structure of the original graph.

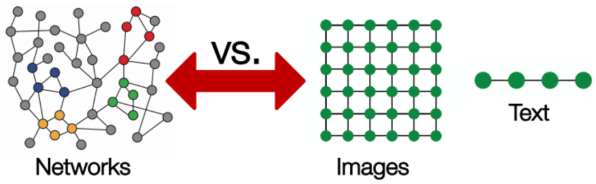


Figure 1: Graph data structure vs conventional data structures.

Recently, GNNs have gained increasing popularity in various domains, including social networks, knowledge graphs, life sciences, and recommender systems. Although GNNs seem to be a promising way of solving graph-related problems, there are still problems associated with their performance, interpretability, and training difficulties. More specifically icily, understanding the difficulty of training GNNs can help to improve the performance of existing GNN models for different types of GNN models. In this paper, our purpose is to study these difficulties by analyzing the effects of activation functions, initialization methods, and dropout rates on training GNNs.

## 2 Motivation

Despite the functional similarities between CNN [1] and GNNs, there are notable differences in the training of GNNs. First, in terms of data, CNN receives structured grid data (image) while GNN models get graph unstructured relational data. Besides, GNNs take input from two domains, the topology domain, and the feature domain (Euclidean).

Secondly, from a topological perspective, as shown in Fig. 2, CNN contains several feature maps, as a result, one neuron/feature might be dropped out for one feature map but still be kept by another feature map. In this way, the information provided by that neuron/feature is still leveraged to construct the next layer in a very robust way with a reduced co-adaption effect. However, in GNN, there is only one feature map and changes in topology structure like dropping out a certain neuron/feature indicate that it does not participate in the optimization process for the subsequent layers at least in the current epoch [2]. In future epochs, although some dropped-out neurons might be re-selected at a future epoch, this is still not good for learning a robust and expressive rep-

resentation. Suppose the model has many layers, then at the last several layers, most of the information will be dropped and message passing becomes inefficient. Therefore, the current dropout-based deep models still perform badly when more layers are added.

Third, for graph data, the dependency structure exists naturally and is modeled by message passing mechanism in GNNs, but in CNNs, the neurons are fully connected manually, and each neuron has many connected neurons, which means that co-adaption inherently exists in GNNs and is stronger. Besides, nodes in GNNs usually have sparse feature vectors, where feature missing is a very common phenomenon. This adds more uncertainty to GNN models [2,3].

Fourth, contrary to deep CNN models, the over-smoothing problem is a notorious phenomenon for GNNs preventing them from going deeper. It is also found that dropout on graph structure or dropout on features may break the message-passing process making the node embedding becomes less expressive to distinguish nodes as the network goes deeper, thus slowing down the pace of over-smoothing.

Finally, Different GNNs such as SCN, GCN, GAT, and ResGCN have different structures and it is possible they will respond differently to different initialization, activation functions, and dropout rates.

Putting all similarities and differences among conventional Deep Learning models especially CNN models with GNNs, it is quite necessary to study the effect of well-known practices in training CNN models in GNNs to get more hints on how can we come up with new models to get more out of current GNN models.

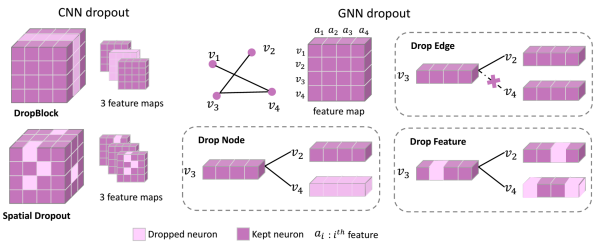


Figure 2: Difference of dropout between CNN and GNN in convolutional layers [4].

## 3 Related Works

One of the first popular GNNs is the Kipf and Welling Graph Convolution Networks (GCN) which is an approach for semi-supervised learning on graph-structured data [5]. As a simpler variant of GCN, Simplified Graph Convolution (SGC) [6] is an another GNN model, removes the non-linear part between GCNs layers to a simple pre-processing step with straight-forward multi-class regression. Bresson and Laurent purpose a model called Residual Gated Graph Convolutional (Res GCN) to solve graph learning problems that are subgraph matching and graph clustering by analyzing the benefit of residuality for graphs [7]. Inspired by the success of the attention mechanism on sequential data Velickovic et al. proposed a graph attention network (GAT) for downstream tasks the most common of which is node classification, learning vector representations for the nodes of a graph [3]. There are other significant research studies and variations of GNNs. Although GNNs have achieved outstanding performance in many tasks, there are still some difficulties.

There are previous studies to understand the difficulties of neural network training. One of the most popular of these is the paper published by Glorot and Bengio [8].In this paper, they discuss many other aspects of training deep networks equipped with sigmoid, hyperbolic tangent, or soft sign activation functions to solve this difficulty. There is a common drawback that training becomes extremely difficult when models become deeper due to more layers would cause noisy information from expanded neighborhoods [5]. Thus, researchers try to find how many layers work best or use residual connection [9] to overcome this issue [5], [3]. Another important point to understand the difficulty of training GNNs is the dropout rate. Dropout [10] is a good regularization method for CNNs. Although GNNs and CNNs share similar model architecture for both with convolutional layers and fully connected layers, the input data structure and convolution operations are different for them. Thus, there is a need for a good understanding of the impact of the dropout rate for more information about the sparse qualities of graphs and the importance of connections among different nodes in the graph data.

## 4 Methodology

Tasks on graph-structured data can be grouped into three levels: node-level, edge-level, and graph-level. Here, we consider the node-level task by which the main goal is training different models using a certain number of labeled nodes to classify unlabeled nodes in a graph. Herein, we aim to investigate the impact of initialization methods, activation functions, dropout rate, and the number of layers on the trained models' performance. To this end, we consider four well-known GNNs, including SGN, GCN, Multi-head GAT (8 heads), and Res GCN, before a fully connected linear layer. We also use Cora, CiteSeer, and Pubmed datasets with 2700, 3327, and 19717 nodes and 1433, 3703, and 500 edges, respectively. As an example, the graphical representation of the Cora dataset is illustrated in Fig. 3, which clearly indicates complex relationships and interdependencies between nodes.

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) [8], and Kaiming Uniform [11]. Next, for dropout analysis, we considered five values including 0.1, 0.3, 0.5, 0.7, and 0.9 which apply before each layer. Finally, to explore the number of layers, we add five GCN convolution layers with 50 node sizes and one fully connected linear layer in two settings: with 50% dropout or without dropout. It is worth mentioning that the selected optimizer is Adam [12] with a learning

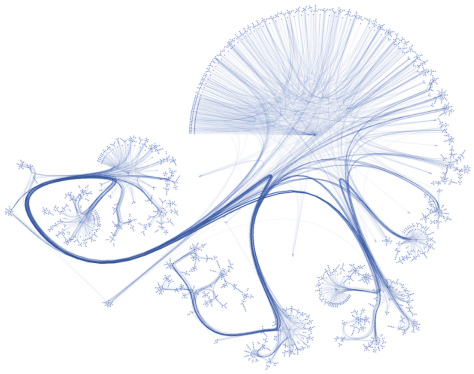

Figure 3: Graph relationships in the Cora dataset.

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.

# 5 Experiment and Results

To understand the difficulty of training GNNs, different case studies are designed by exploring the different numbers of layers, initialization methods, activation functions, and dropout rates. The detailed results for each experiment can be found on the GitHub repository[1].

## 5.1 Case 1: Initialization Methods

A bad initialization can still hamper the learning of a highly non-linear system [11]. In [8] it is concluded that random initialization is doing so poorly with deep neural networks. As a first analysis, the effect of applying different initialization methods on the linear layer of GNN models 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. Fig. 4 shows results for three datasets and four different layers. It can be concluded that as the complexity of the data increase, the effect of the type of initialization methods tends to vanish in the final output and accuracy.

## 5.2 Case 2: Activation Functions

We have tested popular activation functions and investigated their effect on optimization properties in GNNs. Poor activation function can result in slower training time or gradient vanishing problems [11]. For this part of the analysis, different activation functions are applied to the output of each layer (Conv and Linear). The results are illustrated in Fig. 5. 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.

[1] https://github.com/hadiagha/GNNProject

## 5.3 Case 3: Dropout Rate

In [10] it is shown that dropout significantly reduces overfitting and improves the performance and generalizability of CNNs by reducing the co-adaption effect among hidden neurons. Similarly, the co-adaption problem also exists in GNNs, and this effect is propagated and accumulated via message passing among different nodes. Dropout is thus expected to be also useful in GNN models. Fig. 6 demonstrate accuracy results obtained for different datasets and GNN layers as the dropout rate increase. It can be observed that dropping nodes can expectedly decrease the accuracy, but help train a robust and generalizable model. The key finding is 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 regards its counterparts for different dropout rates.

## 5.4 Case 4: Number of Layers

In [8] they found that the logistic sigmoid activation is unsuited for deep networks with random initialization. As a different analysis from previous ones, we investigate the effect of getting deeper with the sigmoid activation function in terms of increasing the number of layers in GNNs. Every 100 epochs, the output of five GCN model layers is extracted and aggregated. Fig. 7 shows the average results for three datasets.

Based on Fig. 7, going deep into more than three convolution layers lead the output to take extremely large values after the 2000th epochs and plummet in accuracy to around 3%. This phenomenon is well studied in [5] and [2]. The main explanation for this phenomenon is that by increasing the number of convolution layers, the embedding model of a convolution layer tends to move from local and neighbor connections into global embedding and in this case, the model losses its discrimination abilities, which is called the over-smoothing problem [4]. Although it is claimed that dropout on graph structure or dropout on features can slow down the pace of over-smoothing, based on Fig. 7, our results do not corroborate this.

# 6 Conclusion

Based on the achieved results, it can be concluded that choices that address locality and neighborhood in graph data, like choosing the type of convolution layer or dropout rate, have a more important effect than choosing initialization methods, activation functions, etc. Because of this important feature of graph data, convolution layers, like GAT, that can learn locality achieved great scores regardless of training configuration choices. These results illustrate that there is great potential for adding localized features to classical deep-learning training configurations like initialization methods or activation functions for graph data.

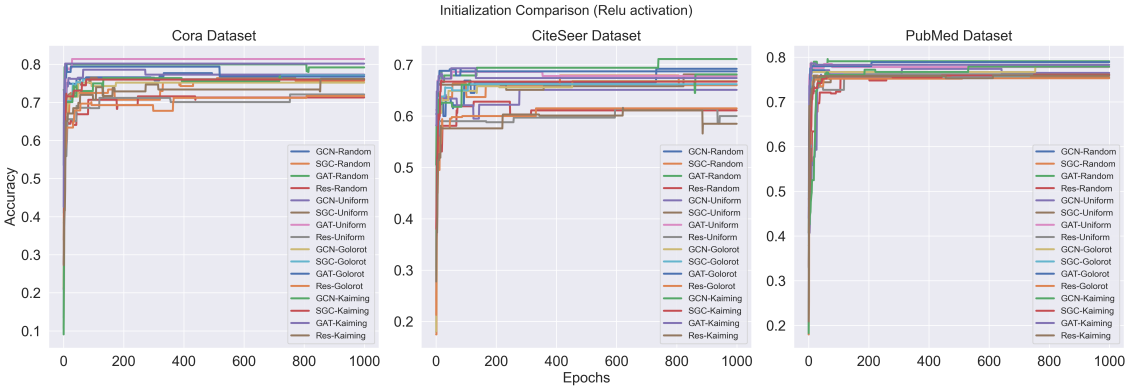


Figure 4: Initialization method comparison for three datasets and four GNN layers.

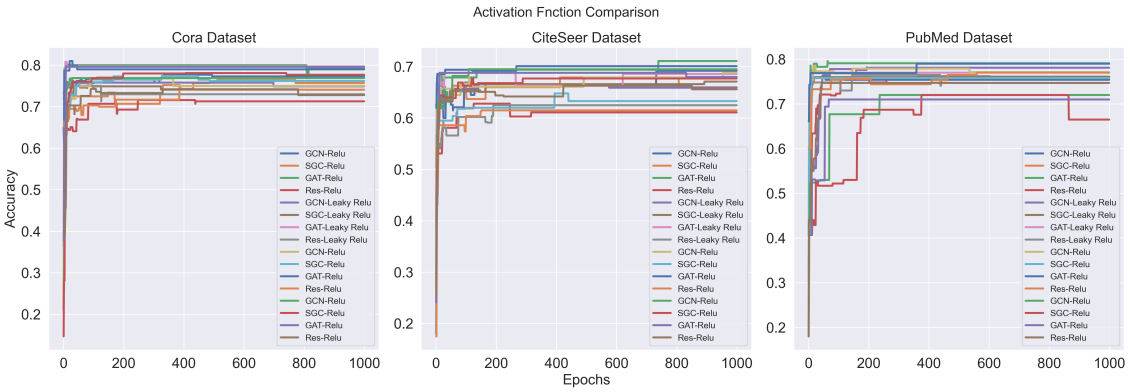


Figure 5: Activation function comparison for three datasets and four GNN layers.

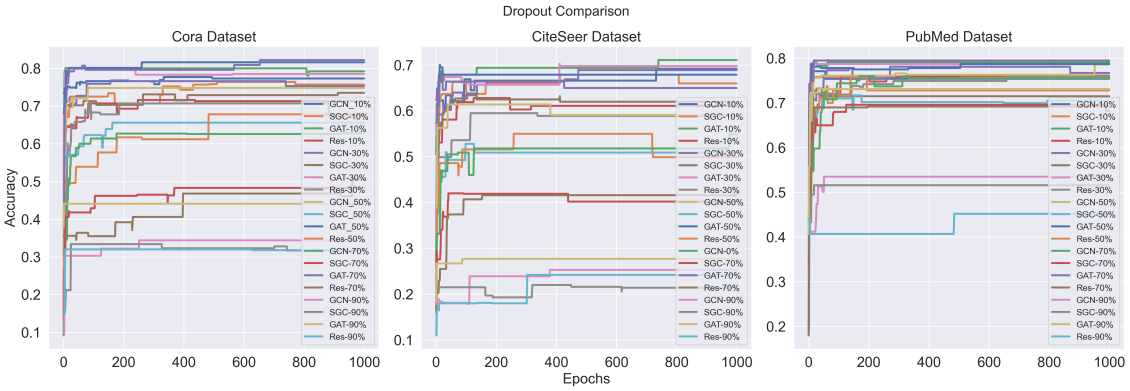


Figure 6: Dropout rate comparison for three datasets and four GNN layers.

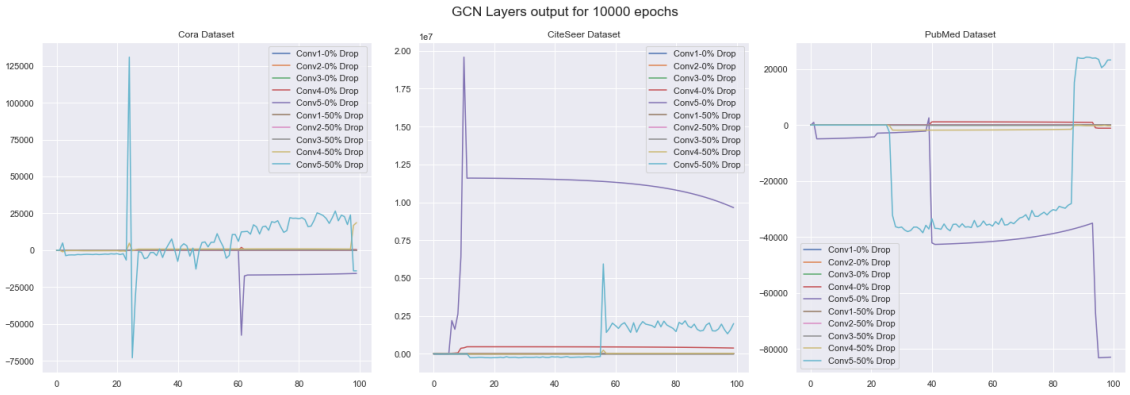


Figure 7: Effect of the number of layers in the GCN model.

# References


[1] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[2] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

[3] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[4] Juan Shu, Bowei Xi, Yu Li, Fan Wu, Charles Kamhoua, and Jianzhu Ma. Understanding dropout for graph neural networks. In *Companion Proceedings of the Web Conference 2022*, pages 1128–1138, 2022.

[5] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[6] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[7] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv: 1711.07553*, 2017.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.