# CSE 676 - Deep Learning
# Project Proposal

**Arthav Ashok Mane**
UBID: 50426537
University at Buffalo
arthavas@buffalo.edu

## Abstract

This document is presented as a report for the final project for the course CSE 676: Deep Learning, taught by Prof. Wen Dong in Fall 2022. Graph Neural Networks (GNN) usually implement Graph Convolution that approximates values by message passing between neighboring nodes. In this project, a recently proposed novel graph diffusion algorithm is implemented that implements a spacially localized graph convolution. The results reproduced from implemented and the results mentioned in paper are found to be similar.

## 1 Introduction

Graph structured data is rife in our everyday life. Graphs are basically connections that shows relations (sometimes with importance i.e. weights) between entities. It can be used to represent concepts and data like social networks, citations, physical models, organizations, transactions, molecules, etc.Graph Learning involves using this graph data for solving tasks like Node classification, Node clustering, Graph classification, and Link prediction.
Just like Convolutional Neural Networks (CNN), we can perform convolutions on layers for graph data as well. This is because images can be considered as graphs of pixels connected throughout multiple dimensions. Each pixel in an image can be considered to be a node with edges to its neighboring pixels in all channels.
Graph convolutions, thus, aggregate values based on immediate neighbors of nodes. However, as an improvement to this, in this project, a recently proposed novel Graph Diffusion Convolution (GDC) algorithm from the paper [1] is implemented. The implemented graph diffusion algorithm is used for the task of node classification for few datasets.

## 2 Graph Diffusion

Graph Neural Networks (GNN) usually implement Graph Convolution that approximates values by message passing between neighboring nodes. However, this limits the networks from looking at the graph as a whole and getting deeper insight into the data. This inadvertently affects the learning and evaluation performance of these networks. In this project, a recently proposed novel Graph Diffusion Convolution (GDC) algorithm from the paper [1] is implemented that uses a spacially localized graph convolution.

Instead of aggregating information from all nodes, GDC aggregates information from a larger neighborhood after sparsifying the data. GDC can be used to preprocess data and can be combined with an form of Message Passing Neural Networks (MPNN) and Graph Neaural Networks (GNN). There are 2 types of Graph Diffusion: * Heat kernel diffusion * Personalized Page Rank (PPR)

## 3 Implementation

### 3.1 Datasets

The paper [1] uses 6 different datasets: Cora, CiteSeer, PubMed, CoAuthor CS, Amazon Computer and Amazon Photo. However, for this project, we only use 3 of these datasets: Cora, CiteSeer and Amazon Photo. Cora and CiteSeer are citation network datasets from [2] where the nodes represent documents and the edges represent citation links. Amazon Photo is a product dataset from [3] where the nodes represent goods and the edges represent the link between two goods that are frequently bought together. Statistics for these datasets is shown in Fig. 1. Furthermore, for preprocessing, the data is modified to only include the largest connected components in the graph.

| Dataset | Nodes | Edges | Features | Classes |
|---|---|---|---|---|
| Cora | 2,708 | 10,556 | 1,433 | 7 |
| CiteSeer | 3,327 | 9,104 | 3,703 | 6 |
| Amazon Photo | 7,650 | 238,162 | 745 | 8 |

Table 1: Dataset Statistics

### 3.2 Models

The following models have been implemented for testing the GDC algorithms for each dataset. All these models are used for node classification.

- Graph Convolutional Network (GCN):
  This network is implemented from the paper [4]. The activation function used for this model is ReLU and the graph convolutional layers have a dropout of probabilty 0.5. For the output, the activation used is Log Softmax.
- Graph Attention Network (GAT):
  This network is implemented from the paper [5]. The activation function used for this model is ELU and the graph attention convolutional layers have a dropout of probabilty 0.5. For the output, the activation used is Log Softmax.
- Jumping Knowledge (JK) Network:
  This network is implemented from the paper [6]. The activation function used for this model is ReLU and the graph convolutional layers have a dropout of probabilty 0.5. For the output, the activation used is Log Softmax.

### 3.3 Training

- Splitting the data:
  The data is split into development and test sets. The development set contains only 1500 nodes whereas the remaining nodes are put into the test set. The development set is further split into train and test set. The training set contains 20 nodes per class whereas the remaining nodes are put into the validation set. The splitting of data is done by modifying the train, val and test masks of the graph data.
- Model Optimization:
  For model optimization, Adam optimizer is used with a learning rate of 0.01. The previous gradients of the layers are cleared. The loss is calculated using the Negative Log Likelihood function. The loss calculated is then back propagated through the layers of the model and the model weights are tuned accordingly.
- Training:
  The training process is run for a maximum of 10,000 epochs. Early stopping patience

is set to 100 epochs. For every run the accuracy is calculated using 100 random splits of development set which is controlled by a set of 100 fixed seeds. Every time a better validation accuracy is obtained, the corresponding results are stored in a dictionary. The hyperparameters used in the training process are listed in Appendix A.

This entire training process is done for all combinations of the three datasets, three models and three diffusion methods.

## 4 Results

The results obtained for the above mentioned implementations are summarized in Table 2. The implementation achieves good accuracies for all combinations. To compare an overview of the performance from the paper [1] and this project's implementation, refer to the Fig. 1 and 2. A direct comparison based on the diffusion method used for each dataset - model combination is shown in the Fig. 3. A comparison of which model - diffusion method achieves the best accuracy fpr each dataset is made in the Table 3. Appendix Sec. B and C show more results.

| Dataset | Model | None | Heat | PPR |
|---|---|---|---|---|
| Cora | GCN | 81.69 ±0.25 | 83.20 ±0.22 | **83.32 ±0.20** |
| | GAT | 80.17 ±0.28 | 79.43 ±0.26 | 80.10 ±0.28 |
| | JK | 81.28 ±0.28 | 82.97 ±0.22 | 83.13 ±0.22 |
| CiteSeer | GCN | 71.99 ±0.31 | **72.53 ±0.30** | 72.39 ±0.25 |
| | GAT | 70.43 ±0.32 | 69.93 ±0.29 | 68.10 ±0.23 |
| | JK | 69.28 ±0.35 | 71.00 ±0.33 | 71.80 ±0.33 |
| Amazon Photo | GCN | 92.04 ±0.21 | 89.24 ±0.26 | **92.67 ±0.21** |
| | GAT | 91.16 ±0.28 | 89.64 ±0.29 | 88.85 ±0.25 |
| | JK | 91.52 ±0.26 | 91.67 ±0.21 | 92.47 ±0.22 |

Table 2: Summary of results (Accuracy %) obtained in the Project

| Dataset | | Accuracy | Model | Diffusion |
|---|---|---|---|---|
| Cora | Paper | 83.78 | JK | PPR |
| | Project | 83.32 | GCN | PPR |
| CiteSeer | Paper | 73.35 | GCN | PPR |
| | Project | 72.53 | GCN | Heat |
| Amazon Photo | Paper | 92.93 | JK | Heat |
| | Project | 92.67 | GCN | PPR |

Table 3: Comparison of Best Accuracies for each dataset

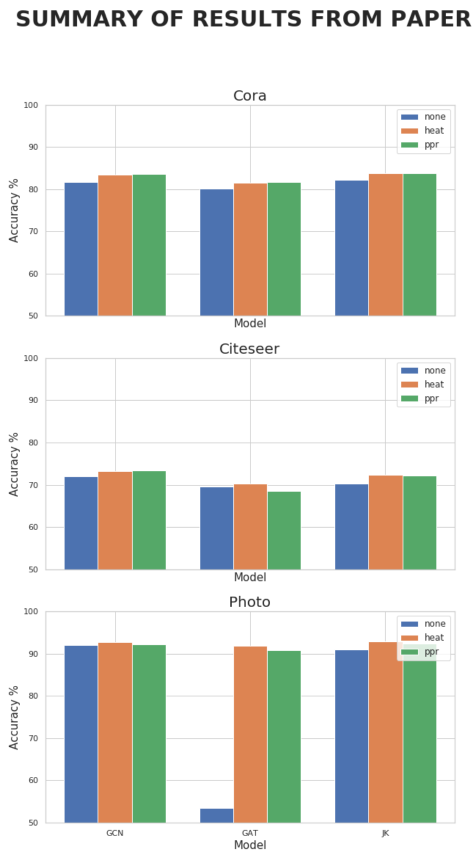


Figure 1: Paper results' summary

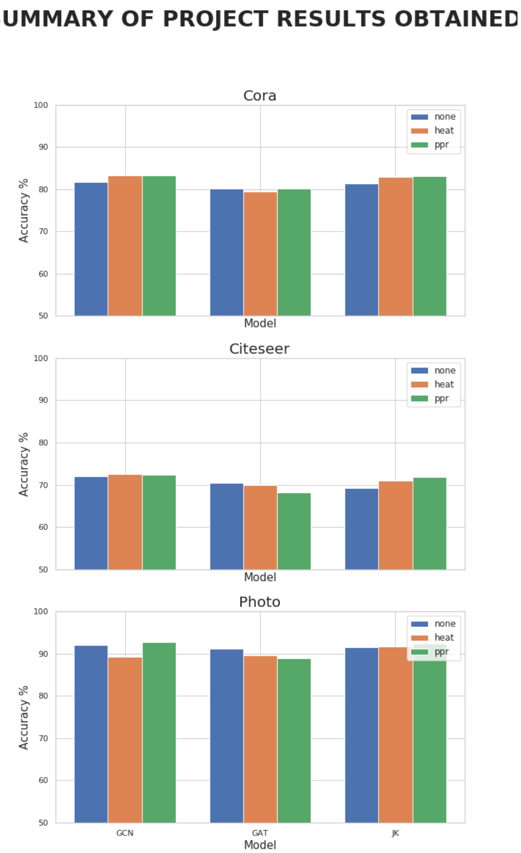


Figure 2: Project results' summary

## 5 Conclusion

GDC is a method based on sparsified graph diffusion which has good performance for node classification. The results produced were almost similar to the ones mentioned in the result. Differences in results might be because of differences in underlying/ backend seeds and hyperparameters. The GDC algorithm is not very scalable. For large datasets, sparse GDC is needed which is only available for the PPR diffusion method currently.

## References

[1] J. Gasteiger, S. Weißenberger, and S. Günnemann. Diffusion Improves Graph Learning. NIPS 2019.

[2] Z. Yang, W. Cohen, and R. Salakhutdinov. Revisiting Semi-Supervised Learning with Graph Embeddings. ICML 2016.

[3] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann. Pitfalls of Graph Neural Network Evaluation. NIPS Workshop 2018.

[4] T. Kipf, and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2017.

[5] P. Veličković, G. Cucurull, A. Casanova, et. al. Graph Attention Networks. ICLR 2018.

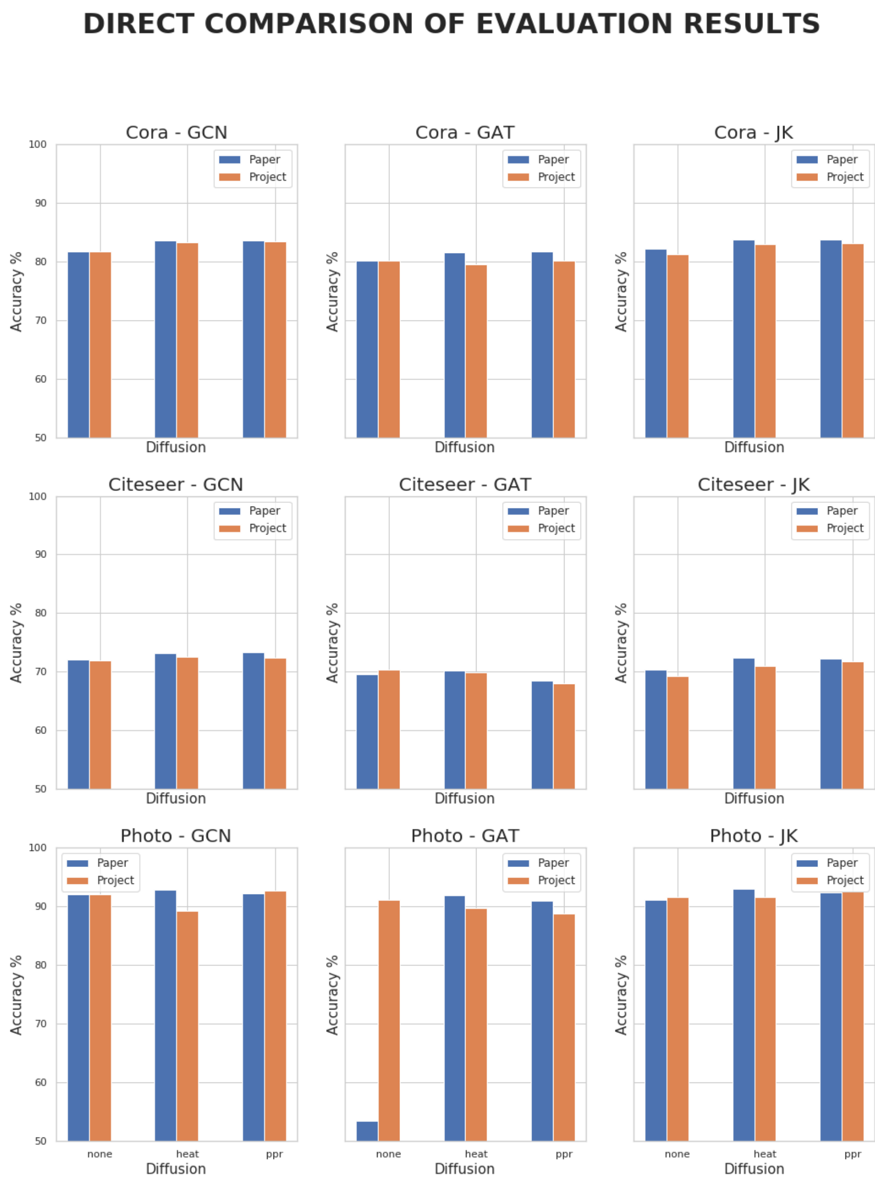


Figure 3: Comparison between results from paper and project

[6] K Xu, C Li, Y Tian, et. al. Representation Learning on Graphs with Jumping Knowledge Networks. ICML 2018.

## A Hyperparameters

The hyperparameters used in training are directly taken from the paper. For all models, Adam optimizer was used with a learning rate of 0.01. All the hidden layers had a dropout of probability 0.5. The remaining model hyperparameters are mentioned in Table 4.

| GCN | | | | | | | |
|---|---|---|---|---|---|---|---|
| Diffusion | Dataset | $\alpha$ | $t$ | $k$ | $\epsilon$ | $\lambda$ | depth |
| None | Cora | - | - | - | - | 0.06 | 1 |
| | CiteSeer | - | - | - | - | 10.0 | 1 |
| | Amz Photo | - | - | - | - | 0.03 | 1 |
| Heat | Cora | - | 5 | - | 0.0001 | 0.09 | 1 |
| | CiteSeer | - | 4 | - | 0.0009 | 10.0 | 1 |
| | Amz Photo | - | 3 | - | 0.0001 | 0.08 | 2 |
| PPR | Cora | 0.05 | - | 128 | - | 0.10 | 1 |
| | CiteSeer | 0.10 | - | - | 0.0009 | 10.0 | 1 |
| | Amz Photo | 0.15 | - | 64 | - | 0.03 | 1 |
| **GAT** | | | | | | | |
| Diffusion | Dataset | $\alpha$ | $t$ | $k$ | $\epsilon$ | $\lambda$ | depth |
| None | Cora | - | - | - | - | 0.06 | 1 |
| | CiteSeer | - | - | - | - | 0.06 | 1 |
| | Amz Photo | - | - | - | - | 0.08 | 1 |
| Heat | Cora | - | 1 | - | 0.0010 | 0.04 | 1 |
| | CiteSeer | - | 1 | - | 0.0010 | 0.08 | 1 |
| | Amz Photo | - | 1 | - | 0.0005 | 0.01 | 1 |
| PPR | Cora | 0.10 | - | - | 0.0050 | 0.08 | 1 |
| | CiteSeer | 0.10 | - | - | 0.0005 | 0.10 | 1 |
| | Amz Photo | 0.10 | - | - | 0.0005 | 0.07 | 2 |
| **JK (with Concatenation)** | | | | | | | |
| Diffusion | Dataset | $\alpha$ | $t$ | $k$ | $\epsilon$ | $\lambda$ | depth |
| None | Cora | - | - | - | - | 0.04 | 3 |
| | CiteSeer | - | - | - | - | 1.00 | 4 |
| | Amz Photo | - | - | - | - | 0.03 | 2 |
| Heat | Cora | - | 5 | - | 0.0001 | 0.09 | 2 |
| | CiteSeer | - | 4 | - | 0.0009 | 1.00 | 2 |
| | Amz Photo | - | 3 | - | 0.0005 | 0.07 | 2 |
| PPR | Cora | 0.05 | - | 128 | - | 0.10 | 2 |
| | CiteSeer | 0.20 | - | - | 0.0009 | 1.00 | 2 |
| | Amz Photo | 0.15 | - | 64 | - | 0.03 | 2 |

Table 4: Training hyperparameters mentioned in paper and used in project

## B Comparison of results for datasets

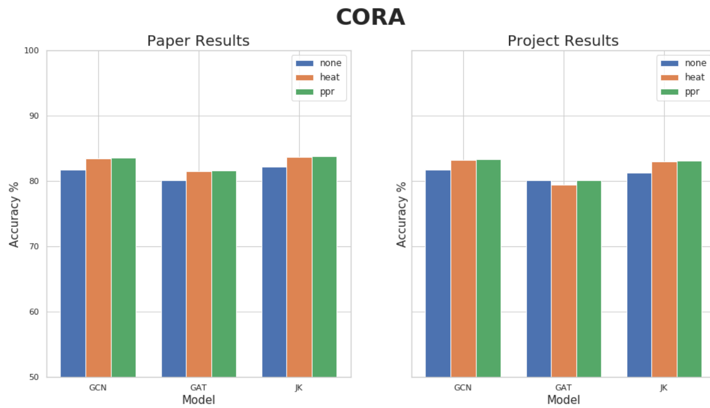


Figure 4: Comparison of accuracies for Cora dataset

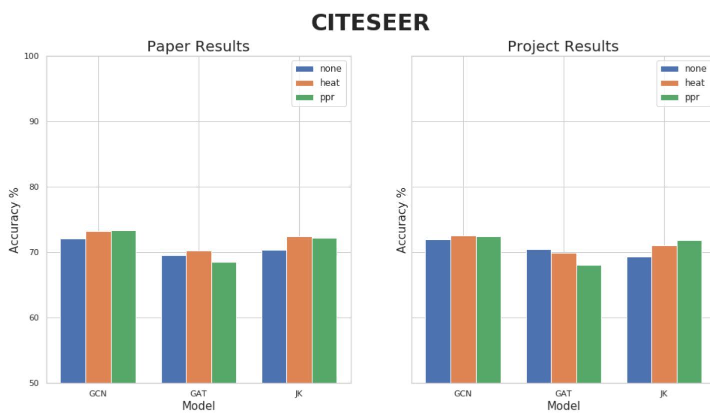


Figure 5: Comparison of accuracies for CiteSeer dataset

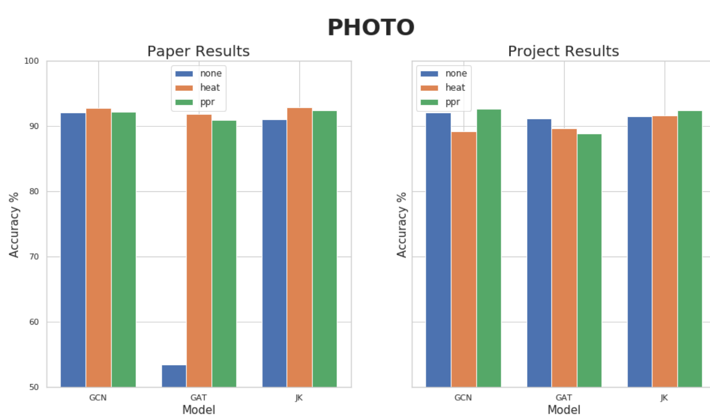


Figure 6: Comparison of accuracies for Amazon Photo dataset

## C Best Accuracies

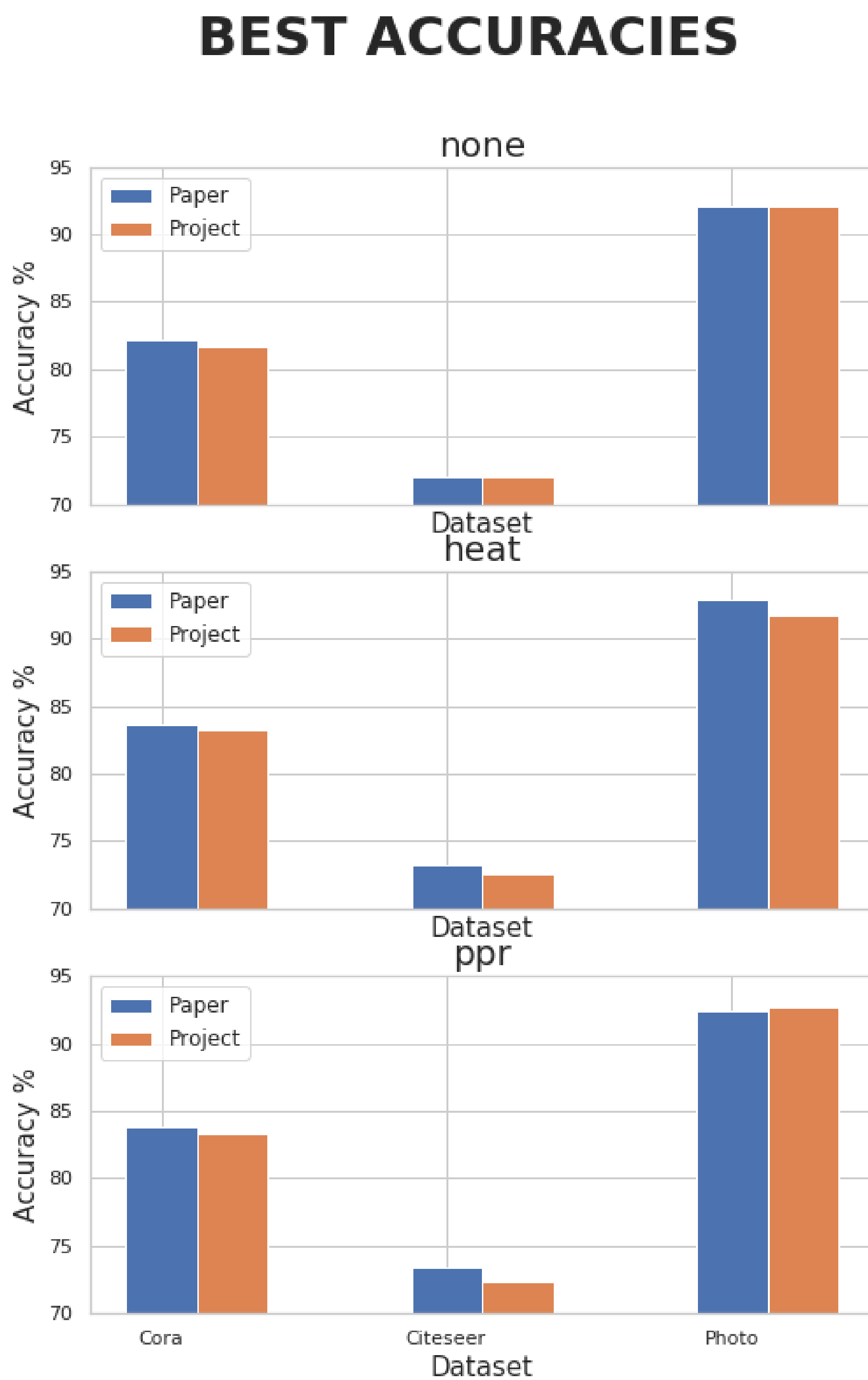


Figure 7: Comparison of best accuracies from paper and project