A study of Graph Neural Networks and Graph Attention Networks for Node Classification

MSc Research Project
Data Analytics

Yash Bhargava
Student ID: x22220861

School of Computing
National College of Ireland

Supervisor: Dr. Giovani Estrada

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Yash Bhargava
Student ID:	x22220861
Programme:	Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Dr. Giovani Estrada
Submission Due Date:	12/08/2024
Project Title:	A study of Graph Neural Networks and Graph Attention Networks for Node Classification
Word Count:	6918
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Yash Bhargava
Date:	12th September 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A study of Graph Neural Networks and Graph Attention Networks for Node Classification

Yash Bhargava
x22220861

Abstract

Graph representations have received considerable attention as they can capture complex data relationships represented as nodes and edges, such as molecular structures, social network analysis, healthcare, and citation networks, etc. Due to this diversity of application areas, finding a suitable network representation to effectively handle complex structures, such as high dimensional term document matrices where the relationship between the nodes and edges is complex is difficult. In this study, the authors have focused on two prominent neural network representations, namely Graph Neural Networks (GNNs) and Graph Attention Networks. These two well-known neural graph architecture are designed to improve the performance of baseline GNN architectures. A range of models are proposed, and were trained and tested on the CORA dataset which consisted of research papers and citations. While the two type of neural models can handle graph based representations, it is not known a priori which one is the most suitable one for node classification. It was found that Graph Attention Networks based on attention mechanisms outperformed all proposed architectures with an accuracy of 73.8%. An accuracy that is even better than the model found in Keras. The major findings of this study show that the graph neural network models based on attention mechanisms were better than the simple GNN node classifiers and also set new targets for effectively handling complex graph-structured data in various applications such as social network analysis and citation networks.

1 Introduction

Recently, Graph Neural Networks have shown some potential advancements in the field of Artificial Intelligence and Deep Learning, mostly for solving tasks that include data in the form of structured graphs. The graph-based models have shown their capabilities over different applications such as in social network analysis, recommendation systems, healthcare domain and many more (Kipf and Welling; 2016). Over the years, one of the most common utilising graphs-based neural networks is for node classification in which we can predict the label of the nodes in a graph based on their features and structure. Node classification can be useful in many research domains such as citation networks where each document can be represented as nodes and classified into various categories based on the content and its citations which are represented as edges in the structured graph. It is difficult for traditional machine learning techniques to capture complex patterns and identify the relationship between the graph-structured data. However, by utilizing the capability of graph neural networks (GNNs) it is easy to effectively extract the features

of nodes and topology of the graph to provide strong node representations which results in improved classification accuracy Wu et al. (2020).
In this study, the author has proposed a novel architecture of the GNN model for the classification of nodes which was focused on improving the performance of graph-based neural networks (GNN) from the baseline architecture of those models. This proposed model has utilized the strengths of existing graph-based models and added some changes to the previous baseline architectures to improve the classification accuracy of the GNN models and overcome some of the limitations of the previous studies. The author evaluates the performance of the proposed architecture of the GNN model on a dataset which consists of research papers with their terms and classifies them into 7 categories of subjects

1.1 Research Question

Graph neural networks (GNN) and graph attention networks (GAT) are two well-known classification techniques for graph based data (data with structural relationship between their entities). However, it is not known a priori whether GNN or GAT are the most suitable techniques for node classification. In order to prove that one technique is better than the other one, a suitable architecture has to be found, which is in itself a major unknown. The question thus becomes:

- What is the best architecture of either GNN or GAT for node classification?

Notice that GNN and GAT are conceptual architectures. The author of this report has to create a network to show that it is indeed better than the other one, as well as find suitable hyperparameters to make it work. A couple of candidate architectures were proposed for both GNN and GAT, and then quantitatively evaluated to find the best architecture for the problem at hand.

1.2 Research Objectives

The main objective of this research is to propose a graph neural network-based model (GNN) that improves the performance for mode classification tasks in the term-document matrix. In the existing baseline GNN models, there were some limitations in that the hyperparameters are fine-tuned for the analysis of complex and high-dimensional datasets which might raise challenges. To overcome these challenges, the study has designed novel GNN and GAT architectures and fine-tuned the hyperparameters of the existing baseline GNN models for the classification of citation networks and more likely to capture complex relationships between the terms and documents. In order to archieve the aforementioned research question, three broad research objectives were carried out:

1. Data preparation & calculation of baseline GNN model (as provided by Keras). Code is publicly available[1].

2. Proposed novel GNN architectures and compared results to the baseline model. Proposed models were fine tuned.

3. Proposed novel GAT architectures and compared results to the baseline model. Proposed models were fine tuned.

[1] `https://keras.io/examples/graph/gnn_citations/`

A detailed experimental setup has been conducted to compare the performance of the proposed architecture with existing baseline models, using performance metrics such as accuracy, precision, recall, and F1-score for the accurate evaluation of the proposed model. By achieving these objectives, the research was focused to significantly improving the state-of-the-art in node classification within term-document matrices.

1.3 Report Structure

The research paper has been divided into many different sections. In the next section 2 of this report, previous studies that have been done on Graph Neural Networks (GNNs) have been studied and analysed. In section 3 the proposed methodology of this study has been discussed, in section 4, the design specification for this research has been discussed. The proposed implementation of this research has been discussed in section 5 and finally, in the last two sections i.e., section 6, how this proposed model has been evaluated and the conclusion of this research in section 7 and lastly the references.

2 Related Work

In this section of the study, we will discuss the various research papers and understand various methodologies used by researchers in the study of Graph Neural Networks (GNNs). This section of the study is divided into the following subsections: subsection 2.1, subsection 2.2, subsection 2.3, subsection 2.4, subsection 2.5.

2.1 GNNs in Healthcare and Biology

In the research (Ramirez et al.; 2020), the research presented four models based on a graph convolutional neural network in which the unstructured gene expressions were given as inputs for the classification of 33 distinct types of cancer or normal. The four different GCNN models were trained on the co-expression graph, co-expession+singleton graph, protein-protein interaction graph (PPI) graph and singleton +PPI graph. The models were trained on a large set of 10,340 cancer tissue samples and 731 normal samples of the TCGA dataset and the results of these models reveal excellent performance on these datasets of 94.7% among these 34 classes. The results of these 4 models were excellent in the classification of cancer types or normal tissue and achieved an accuracy of more than 94%. However, there were some limitations as well such as the generalizability of the model which should be tested on independent datasets and the quality of data used was high-quality gene expression data which enables these models to be limited to that only. The study (Monroy et al.; 2024) proposed a novel graph-based method for analysing histopathological images, for the classification of various cancer types. The method utilizes cell-graph representations, which combine the spatial organization and interactions of cells to improve classification accuracy. This graph-based representation outperformed the traditional methods which showed the robustness of GNNs in multi-class classification. The author also introduced a novel graph-based approach for the classification of multiple diseases. However, there are some limitations as well such as the complexity of the methodology that might arise challenges for the implementation in clinical settings. Also, the effectiveness might differ on the quality of histopathological datasets. The study (Bera et al.; 2024) presented PND-Net, which was a deep network designed for recognizing plant nutrition deficiencies by using a Graph Convolutional Network (GCN) module keeping the CNN as a backbone of the architecture. The model was evaluated on four datasets showing various plant nutrition deficiencies and leaf diseases, showing its effectiveness in real-world agricultural settings. The study showed the high productivity of the model for the classification of various plant nutrition deficiencies over multiple datasets which provides transparency and reproducibility. However, there were some limitations such as the integration of GCN with CNN increased computational complexity and model size. The study (Chen, Zhang, Wang, Zekelman, Cetin-Karayumak, Xue, Zhang, Song, Makris, Rathi et al.; 2024) presented a hybrid model that integrated Graph CNN and Transformer architectures for machine learning with diffusion MRI tractography. The proposed model captured local and global features by making use of anatomical relationships and long-range interactions, indicating the strong performance in a sex prediction task over two large datasets. The study has some advantages such as using the hybrid approach which combined the strengths of GCNNs and Transformers which have improved the feature extraction and classification accuracy. The study has incorporated local anatomical information and global feature dependencies, which resulted in improving prediction accuracy. However, there are some limitations as well such as the model is specifically for the diffusion MRI tractography, which limits the applicability to other imaging modalities and the hybrid architecture requires enough computational resources and knowledge for the implementation of these approaches.

2.2 GNNs in Technical and Industrial Applications

The study (Jin; 2024) presented a graph-based convolutional neural network (GCNN) model named Graph-CNNpred which was developed to predict the stock market trends for indices such as S&P 500, NASDAQ, DJI, NYSE, and RUSSELL. The model combines various range of data sources to improve prediction accuracy. The study results showed that Graph-CNNpred outperformed all the other state-of-the-art baseline algorithms by 4% to 15% in terms of F-measure. The proposed model has some advantages that it showed improvements in the performance from the other baseline models, and it has the potential to be implemented in real-world applications. However, this research has some limitations as well such as the architecture of the model was complex and needs enough computational resources for the implementation and the study was only focused on the directional movement of the stock and ignoring the other market dynamics. The study (Lu et al.; 2024) has used Graph Neural Networks (GNNs) for wireless networks, which were focused on graph representation, its architecture, and the evaluation of the performance. The study evaluated various GNN models, like GCN, GAT, and RGAT, for several tasks such as signal-to-interference-plus-noise ratio (SINR) prediction and allocation of power. The results of the study have shown that GNN-based models offer better performance as compared to traditional methods. Mainly, both supervised and unsupervised learning approaches provided results which were close to optimal and showed the effectiveness of the GNNs models for the optimisation of wireless networks. However, there were some limitations as well such as the proposed model was only explored within wireless networks, which limits the generalizability of the model. The study (Chung et al.; 2024) identified the use of Graph Neural Networks (GNNs) for predicting the effective elastic moduli of rocks, which were important for various geophysical and engineering applications. The study made graphs of the rocks microstructure and implemented GNNs for capturing the complex relationships between rock components. The results of the proposed model showed improved accuracy over traditional methods and improvement in predictive performance. However, the study has some limitations as well such as the model generalizability as it was only focused on prediction of rock elastic moduli which might limit the applicability of the findings to other geophysical properties and the model requires microstructural data which was not available easily. This study (Dash et al.; 2024) proposed a hybrid Fault Detection and Isolation (FDI) method that combines the strengths of Bond Graph (BG) and Convolutional Neural Network (CNN) to improve the overall FDI performance with a minimal number of labelled data. The proposed model employed the residuals generated from the BG model to improve fault isolation. The study used a simulated model of a Direct Current (DC) motor to generate datasets for training and testing the proposed model. The datasets include various types of faults, including incipient and step faults, as well as multiple simultaneous faults. The results show that the proposed BG-CNN method outperforms traditional machine learning algorithms, such as Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbour (KNN), and Artificial Neural Network (ANN), in terms of F1-score, especially when the number of labelled data is limited. The study has several advantages i.e., the proposed model only requires lass but labelled data to get good results and could effectively deal with incipient faults and isolate multiple simultaneous faults. However, there were some limitations, such as the computational cost of the model due to the use of CNN and not suitable for systems with highly nonlinear dynamics.

2.3 GNNs in Computer Vision & Natural Language Processing

The study (El-Gayar et al.; 2024) proposed a method to detect deep fake videos by utilizing the potential of Graph Neural Networks (GNNs). The study constructed a graph where each node represented a video frame, and edges were encoded as temporal and spatial relationships between frames. The study resulted GNN-based model achieving higher accuracy for the detection of deep fakes as compared to the traditional CNNs and RNNs. The proposed model showed good results for different datasets, which consisted of high-quality deep fakes that were challenging to detect. The study proposed a novel application of GNNs in the domain of video analysis and deep fake detection. However, there were some limitations as well such as GNN-based models can be computationally expensive, as they require enough resources for both training and testing and processing very large video datasets may show some scalability issues. This study (Chen, Xiao, Du, Zhao, Zhang, Wu, Zhu, Zhang, Yao, Hu et al.; 2024) proposed a unified and biologically plausible framework for understanding the Vision Transformers (ViTs) by making relational graph representations. The relational graph consisted of two components: an aggregation graph, representing spatial interactions between network channels, and an affine graph, representing information communication within the network. The study results identified that the performance of ViT models was correlated with certain graph measures like clustering coefficient and average path length. The study implemented this model to optimize the aggregation graph led to better performance of ViTs. However, there were some limitations of this study as well such as this proposed framework was not generalizable to all Vision transformers (ViT). The study (Kouris et al.; 2024) proposed a methodology for text summarization that utilizes semantic graphs for representing the content of text documents. In this study, nodes in the graph represented the key concepts, while edges represented the semantic relationships between these concepts. The study employed a semantic graph-based method that produced summaries that were more informative and better preserved the semantic meaning of the original text as compared to traditional summarization techniques. The methodology outperformed baseline models in various metrics such as ROUGE scores, which show higher-quality summaries. However, there were some limitations of this study as well such as the construction and processing of semantic graphs requires enough computational resources, which limits the scalability of the methodology, and this methodology could also face challenges with large and complex documents as it requires more refinement to handle effectively. The study (Gao et al.; 2024) presented GraphormerDTI, a graph transformer-based approach for the prediction of drug-target interaction (DTI). The authors have made use of the Graph Transformer neural network for modelling molecular structures and embedding molecular graphs into vector-format representations through repetitive Transformer-based message passing. The proposed model aims to effectively make informative representations for out-of-sample molecules, which enables DTI prediction over molecules with exceptional performance. The proposed model GraphormerDTI has outperformed all five state-of-the-art baselines for out-of-molecule DTI prediction. However, there were some limitations as well i.e., the study only considered three benchmark datasets, which might not fully capture the complexity and variability of real-world DTI data.

2.4 Recent Advancements with GNNs

This study (Shi et al.; 2024) explored the relationship between traditional Convolutional Neural Networks (CNNs) and Graph Convolutional Neural Networks (Graph CNNs). The study proposed the "companion model," which used a companion graph and signal representation to demonstrate that traditional CNNs can approximate Graph CNNs under specific conditions. This approach used Graph Signal Processing (GSP) to adapt traditional CNN layers for graph data, which shows that traditional CNNs can perform similarly to Graph CNNs for graph classification tasks. However, there were some limitations as well such as the study relied on specific conditions, i.e., the uniqueness of eigenvalues, which might not hold for all graph structures and in this study, the companion model might have some complexity to the understanding theoretical and implementation of traditional CNNs for graph data. This study (Besta et al.; 2024) was focused on the improvement of graph representation learning by combining higher-order structures and dynamic graph evolution. The authors have proposed using efficient transformers for capturing complex relationships and temporal changes in graphs. The study utilized efficient transformers which handled the large-scale data with reduced computational resources. However, there were some limitations as well such as the approach's scalability to very large graphs and long-term dynamics requires further evaluation. The study (Kofinas et al.; 2024) proposed the use of Graph Neural Networks (GNNs) for learning equivariant representations, which were the representations that remain invariant under some transformations. The study has utilized the structural properties of graphs to develop GNNs that could effectively learn and generalize these equivariant representations for various tasks. The study applied to the various tasks that might benefit from the equivariant representations. However, there were some limitations as well such as the proposed approach might require some adjustments for the existing GNN frameworks for the learning of equivariant representations. The study (Li et al.; 2019) explored the possibility of training deep Graph Convolutional Networks (GCNs) by utilizing the concepts from Convolutional Neural Networks (CNNs) such as residual/dense connections and dilated convolutions. The authors have proposed various techniques to train deep GCNs by adapting residual/dense connections and dilated convolutions from CNNs to GCNs. The authors have shown the impact of using these deep GCN frameworks by building a very deep 56-layer GCN and showing how it significantly improves the performance in the task of point cloud semantic segmentation. The authors have also discussed the benefits of their work for advancing GCN-based research. However, there were some limitations as well such as the study does not provide a thorough comparison with other state-of-the-art GCN models, making it difficult to fully evaluate the performance of the proposed approach. The study (Ning et al.; 2024) proposed a model which has utilised Graph Neural Network (GNN) for forecasting Sea Surface Temperatures (SST) and Sea Surface Temperature Anomalies (SSTA) on a global scale. The authors have proposed an improved graph construction technique for SST teleconnection representation and shown the capability of the Graph SAGE model for 1-month-ahead global SST and SSTA forecasting. The GNN model outperformed both the persistence model and traditional methods for SST and SSTA forecasting. The study has revealed the potential of GNNs in climate forecasting, for understanding the spatial patterns and interconnections between different oceanic regions. The proposed Graph SAGE model forecasted SSTs up to 2 years ahead and SSTAs 1 month in advance, with improved accuracy in regions with strong ocean currents. The findings of the study show that using appropriate graph re-sampling and GNNs could help understand the complex climate system.

2.5 Conclusion of the literature review

The review of these studies has shown the improvements and applications of Graph Neural Networks (GNNs) in various domains like healthcare, biology, technical, industrial, computer vision, and natural language processing domains. These applications have shown the flexibility and robustness of GNNs, as scalability and computational complexity remain challenges. In this research, the author focussed on graph representations in Convolutional Neural Networks (CNNs) and compared the proposed architecture with baseline GNN architectures for node classification tasks. This research goal is to evaluate the effectiveness of combining graph-based methods in CNNs and to identify important improvements as compared to the baseline architectures.

3 Methodology

In this section, the author has discussed the methodology that has been used in this study for node classification tasks on research paper and citations dataset. The author has utilized a Cross-industry standard process for data mining (CRISP-DM) methodology to proceed with this study. This methodology has been broken down into six steps. The structure of this methodology can be seen in Figure 1

- **Business Understanding**: Graph Neural Networks (GNNs) have become an important part of artificial intelligence and deep learning in the last few years, this is only used for structured graph data problems. They are used in many domains like social network analysis, recommendation systems or healthcare. Usually, GNNs have been applied to node classification tasks in which the main goal is to predict labels of nodes in a graph based on their features and relationships.

- **Data Understanding**: The dataset consists of research papers which are shown in a term-document matrix that contains attributes of the papers and citations. This

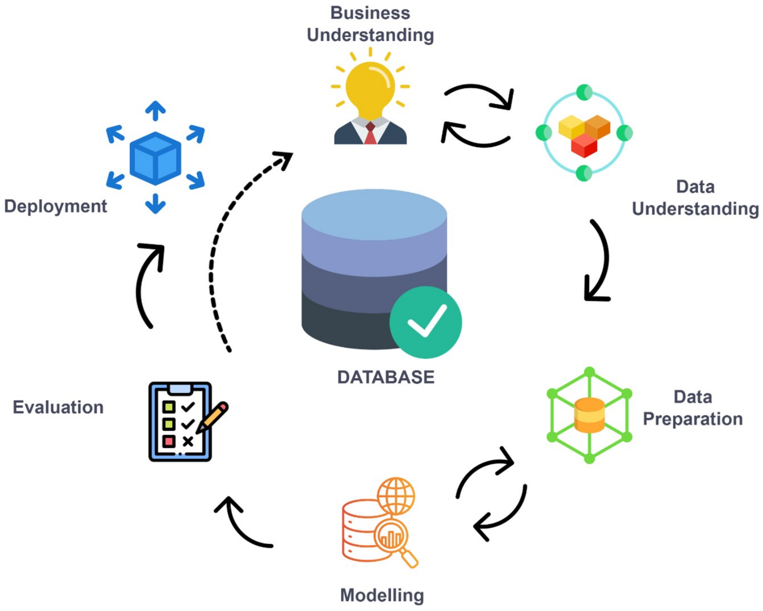


Figure 1: CRISP-DM Methodology

data is structured as a graph where nodes represent documents and edges represent citations. The dataset has seven different classes: Case-Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning and Theory. Firstly exploration involves finding node edge distribution patterns, identifying patterns and looking for missing values. In this stage, we are trying to obtain some ideas about the structure of data and its properties.

- **Data Preparation**: This is the third step which is very important for the modelling process. First, perform the Normalization which means Normalizing the term-document matrix to ensure uniformity in feature scaling. After this the Graph Transformation where the author Convert the term-document matrix into a graph format suitable for GNNs. This involves creating nodes for each document and edges for citations. The last step is to Split the graph data into training and test sets to ensure robust model evaluation.

- **Modelling**: There are two types of architecture used GNN Node Classifier Architecture 1 which is built on Basic architecture with predefined hyperparameters and GNN Node Classifier Architecture 2 which includes some advanced layers and features to increase the performance and accuracy. The second architecture GAT Node Classifier Architecture 1 Utilizes multi-head attention mechanisms and there is also a GAT Node Classifier Architecture 2 which uses Refined attention-based architecture with different hyperparameters. There are various hyperparameter techniques such as grid search or random search to fine-tune hyperparameters, including the number of hidden units, dropout rates, types of aggregation and combination, and normalization are used to increase the performance and accuracy.

- **Evaluation**: In this part, the author trains the model on the training set ensuring that the process includes validation to prevent overfitting. After this Evaluate both models using different evaluation metrics such as Accuracy, Precision, Recall, and F1-Score. Compare the performance of the proposed models against existing baseline models. Highlight the improvements achieved in node classification accuracy and other performance metrics.

- **Deployment**: Test the data on the testing data on the final model architecture using appropriate tools and programming languages. Ensure that implementation can handle datasets effectively and can be replicated for future research. Present detailed architecture of final models, including several trainable and non-trainable parameters. Also, provide accuracy and loss curves which would help to understand how well it performs in future.

4 Design Specification

In this section of the paper, the author has mainly focused on describing the design specification of this research such as how the flow of the research went through and what are the design specification of this research work.

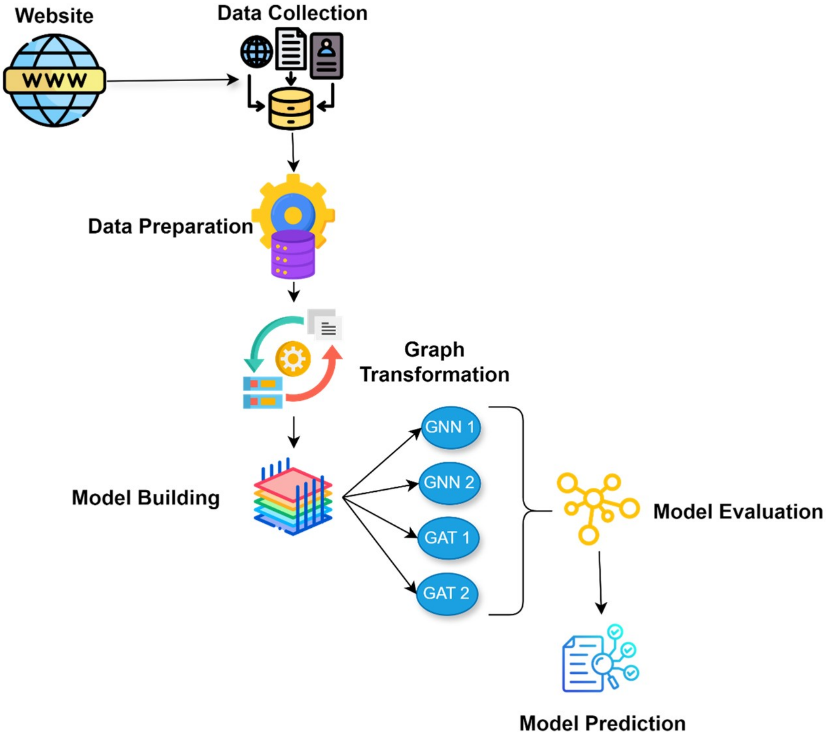


Figure 2: Design Specification of the research

Firstly, the dataset CORA.content contains 2708 records of papers and 1435 columns, and the second dataset CORA.citations of citations contains 5429 records and 2 columns were fetched from the website and loaded into the jupyter notebook. Then the dataset was preprocessed and the structured graphs were constructed on these pre-processed data for the transformation of these term-document matrices into graph format so that this transformed data can proceed with the next step i.e., the model-building stage in which this report is looking at various proposed graph neural network based node classifiers. In this study, the author has proposed 4 different architectures two architectures of GNN Node Classifiers and two architectures of Graph Attention Networks (GAT) with attention mechanisms and also tried with different hyperparameters then these models were compared with each other and baseline architectures that were proposed previously. Finally, the proposed architectures performed better than the baseline architectures in terms of their performance, accuracy and the complexity of the model. In the end, the proposed architectures have made predictions on the randomly generated instances and evaluated their test accuracies.

5 Implementation

In this section of the report, we will discuss the tools and languages that were used while carrying out this research which was aimed to improve the accuracy of the baseline Graph neural Networks (GNN) models on the citations and terms documents dataset for node classification of seven classes of subjects based on their feature vectors. The final implementation stage focused on graph transformation, model development, training and evaluation of the proposed architectures of GNNs.

In the study, firstly the author pre-processed the data and transformed the term-document matrix data into graph-based structures to provide an input to the graph-based models for the node classification tasks which basically involved the normalization of the data and then converting it into graph-based structures as we can see in Figure 3 After the data transformation into a suitable structured graph format, the author proceeded with the model-building stage and randomly split the graph data into training and test sets. The author has a user-defined function for building the model such as a function for creating a feed-forward network function which was used for preprocessing of the graph data, a function for defining the Graph Convolutional layer (GraphConvLayer) for determining

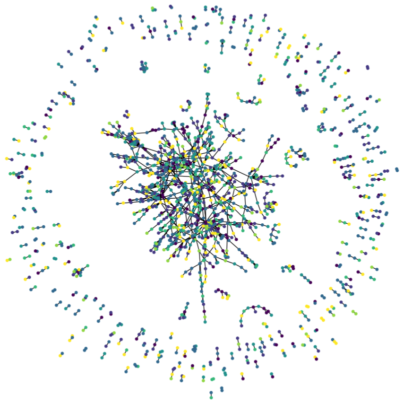


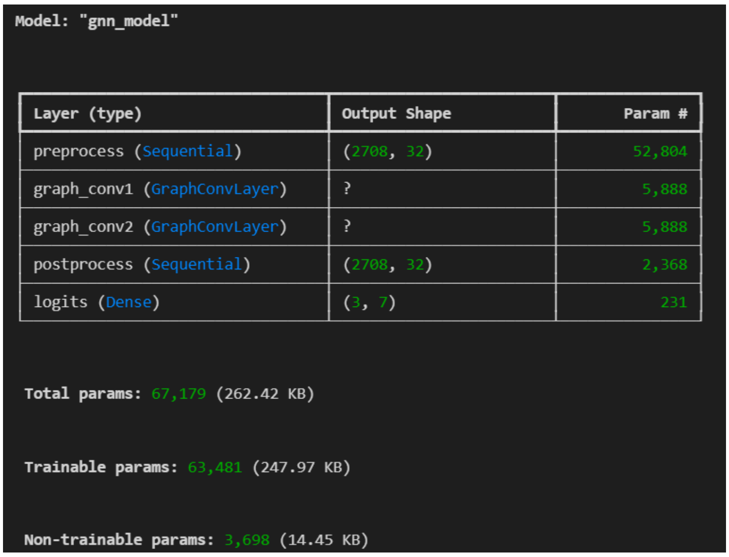
Figure 3: Normalized graph structure

the type of layer, a function for building the GNN Node Classifier model in which the author has used these functions to build the custom Node Classifier model, function for the custom Graph Attention Networks model with suitable attention mechanisms.

The structural difference exists between the GNNs and GATs by how they combine several features of information from the neighbouring nodes. In this study, there was a fixed type of aggregation used for combining the feature vectors from the nodes and the weights of each node were considered as equal or it used the predefined weights of the edges. In GATs, which uses the multiple head attention layers that uses the attention mechanism which means that different weights for each neighbouring nodes were considered which allows the GAT models to give priority to important nodes of significant feature vectors that results in getting more optimised and flexible way to combine the feature vectors.

5.1 Baseline Architecture

A graph-based neural network for node classification tasks was developed by the Keras team which acts as a baseline architecture for this study[2]. The architecture of this baseline graph neural network model (GNN) is very simple. Its structure contains only 2 Graph Convolutional layers with 32 neurons in each layer, with a learning rate of 0.01, a dropout rate of 0.5 and a batch size of 256 was trained for three hundred epochs. While fitting the model callbacks as early stopping were specified to prevent overfitting. The baseline model contains feed-forward networks for the preprocessing and post-processing of inputs and outputs of the last layer and has 63,481 trainable and 3,698 non-trainable parameters. This work covers Research Objective 1. The summary of this baseline architecture can be seen in the Figure 8.



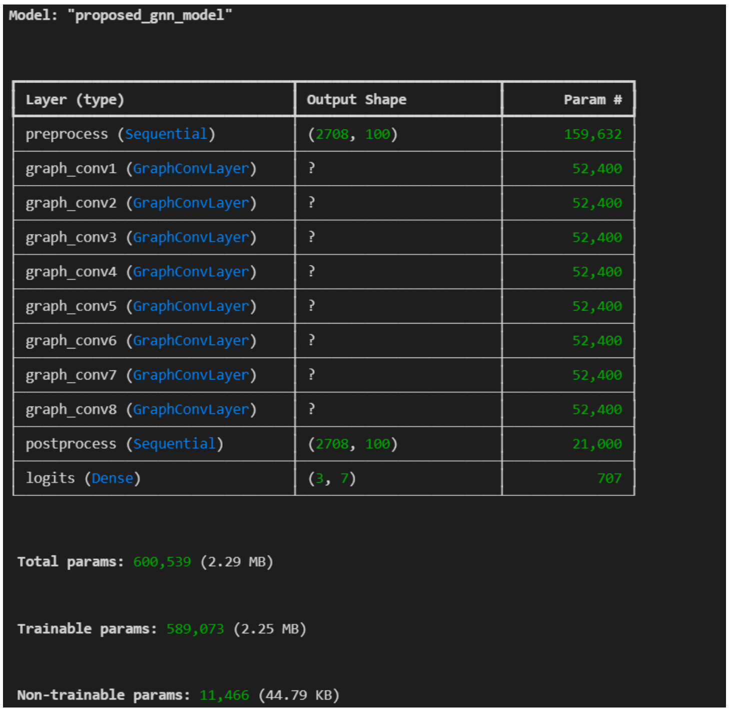
```
Model: "gnn_model"


┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape             ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ preprocess (Sequential)          │ (2708, 32)               │        52,804 │
├──────────────────────────────────┼──────────────────────────┼───────────────┤
│ graph_conv1 (GraphConvLayer)     │ ?                        │         5,888 │
├──────────────────────────────────┼──────────────────────────┼───────────────┤
│ graph_conv2 (GraphConvLayer)     │ ?                        │         5,888 │
├──────────────────────────────────┼──────────────────────────┼───────────────┤
│ postprocess (Sequential)         │ (2708, 32)               │         2,368 │
├──────────────────────────────────┼──────────────────────────┼───────────────┤
│ logits (Dense)                   │ (3, 7)                   │           231 │
└──────────────────────────────────┴──────────────────────────┴───────────────┘


 Total params: 67,179 (262.42 KB)


 Trainable params: 63,481 (247.97 KB)


 Non-trainable params: 3,698 (14.45 KB)
```

Figure 4: Model Summary of Baseline Architecture

[2] https://keras.io/examples/graph/gnn_citations/

5.2 GNN Node Classifier Architecture 1

A graph-based GNN Node Classifier model was proposed in this study in which the architecture takes parameters as graph_info, the number of classes (num_classes), number of hidden units or neurons in a convolution layer, type of aggregation, kind of combination, dropout rate, normalization is done or not. The first proposed GNN Node Classifier consists of a sequential model for preprocessing before passing it to graph convolutional layers which were equal to 8. Then the output of these convolutional layers 8 is again passed to the sequential model for post-processing the output and then a dense layer for the prediction of logits for node classification. This was the whole architecture of our proposed GNN Node Classifier which includes 589,073 trainable parameters and 11,466 non-trainable parameters as can be seen in the model summary in Figure 5.



```
Model: "proposed_gnn_model"

┃ Layer (type)                 ┃ Output Shape       ┃ Param # ┃
│ preprocess (Sequential)      │ (2708, 100)        │ 159,632 │
│ graph_conv1 (GraphConvLayer) │ ?                  │  52,400 │
│ graph_conv2 (GraphConvLayer) │ ?                  │  52,400 │
│ graph_conv3 (GraphConvLayer) │ ?                  │  52,400 │
│ graph_conv4 (GraphConvLayer) │ ?                  │  52,400 │
│ graph_conv5 (GraphConvLayer) │ ?                  │  52,400 │
│ graph_conv6 (GraphConvLayer) │ ?                  │  52,400 │
│ graph_conv7 (GraphConvLayer) │ ?                  │  52,400 │
│ graph_conv8 (GraphConvLayer) │ ?                  │  52,400 │
│ postprocess (Sequential)     │ (2708, 100)        │  21,000 │
│ logits (Dense)               │ (3, 7)             │     707 │

 Total params: 600,539 (2.29 MB)

 Trainable params: 589,073 (2.25 MB)

 Non-trainable params: 11,466 (44.79 KB)
```

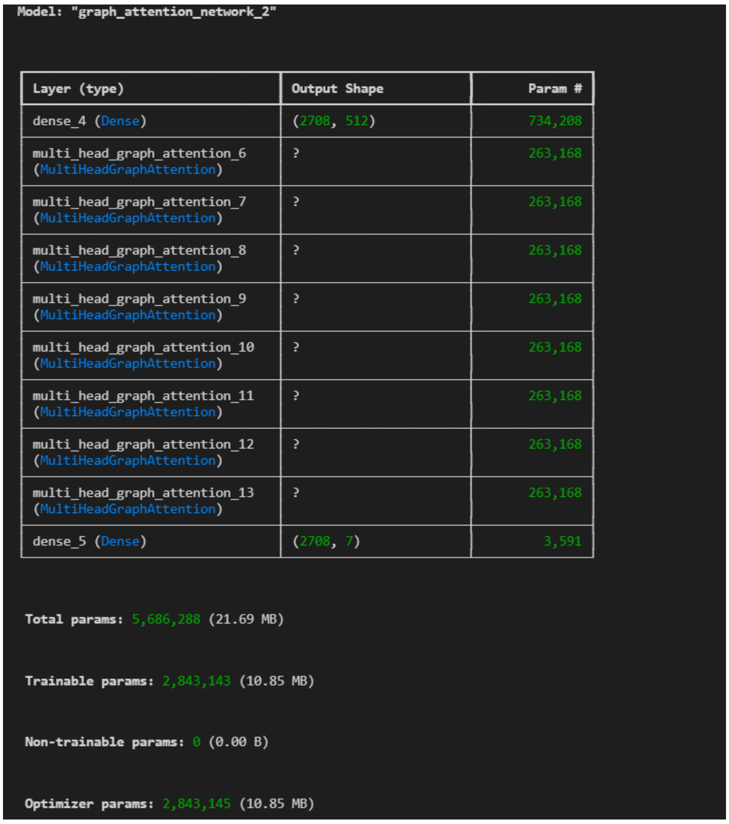
Figure 5: Model Summary of GNN Architecture 1

5.3 GNN Node Classifier Architecture 2 with tuned hyperparameters

A second graph-based GNN Node Classifier proposed Architecture was built by tuning some of the hyperparameters such as number of hidden units, dropout rate, type of aggregation, type of combination and Boolean normalize value for number of maximum 15 trials and found out the best hyperparameters as number of hidden units as [88, 96], dropout rate as 0.4, aggregation type is sum, combination type as add and normalization setting remains False. This model was built on the same number of Graph Convolutional layers as of the previous proposed architecture i.e., 8. This proposed model was built using above above-tuned hyperparameters which include trainable parameters and non-trainable parameters.

5.4 Graph Attention Networks (GAT) Architecture 1

The third graph-based Node Classification model proposed was built by utilizing the attention mechanisms in which the author has used "MultiHeadGraphAttention" layers which were built using Attention mechanisms. The author used several hyperparameters such as the number of hidden units as 64, the number of heads as 8, the number of hidden layers as 8, and batch size as 32 and then the model was trained on the data for 50 epochs. This was the whole architecture of our proposed Graph Attention Networks (GAT) Node Classifier which includes 2,843,143 trainable parameters and 0 non-trainable parameters as can be seen in the model summary in Figure 6. In this architecture, the optimizer that was utilized was Stochastic Gradient Descent (SGD) with a momentum of 0.9 and a learning rate of 0.01.

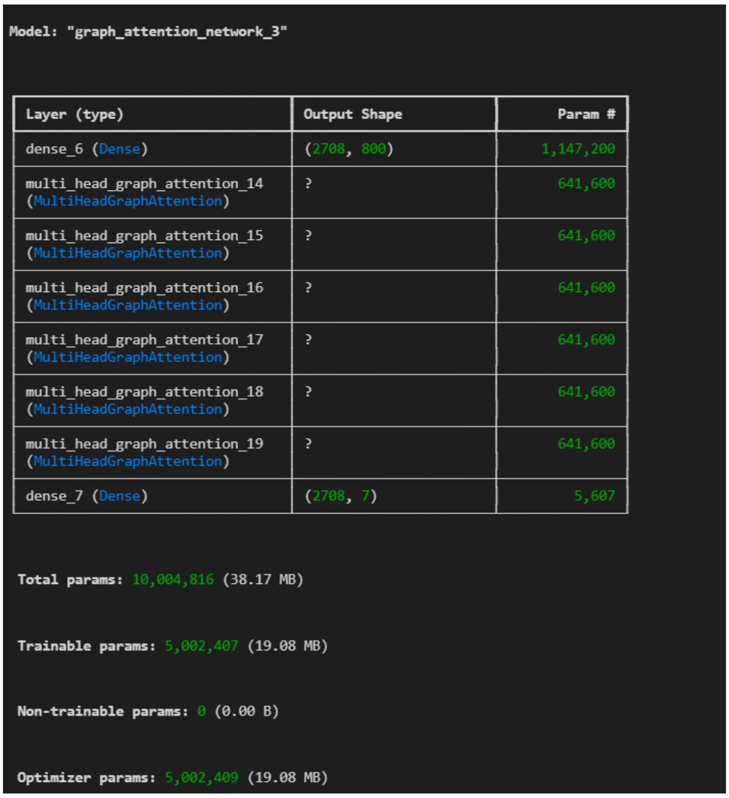


```
Model: "graph_attention_network_2"

┌──────────────────────────────────┬──────────────────────────┬────────────────┐
│ Layer (type)                     │ Output Shape             │        Param # │
├──────────────────────────────────┼──────────────────────────┼────────────────┤
│ dense_4 (Dense)                  │ (2708, 512)              │        734,208 │
├──────────────────────────────────┼──────────────────────────┼────────────────┤
│ multi_head_graph_attention_6     │ ?                        │        263,168 │
│ (MultiHeadGraphAttention)        │                          │                │
├──────────────────────────────────┼──────────────────────────┼────────────────┤
│ multi_head_graph_attention_7     │ ?                        │        263,168 │
│ (MultiHeadGraphAttention)        │                          │                │
├──────────────────────────────────┼──────────────────────────┼────────────────┤
│ multi_head_graph_attention_8     │ ?                        │        263,168 │
│ (MultiHeadGraphAttention)        │                          │                │
├──────────────────────────────────┼──────────────────────────┼────────────────┤
│ multi_head_graph_attention_9     │ ?                        │        263,168 │
│ (MultiHeadGraphAttention)        │                          │                │
├──────────────────────────────────┼──────────────────────────┼────────────────┤
│ multi_head_graph_attention_10    │ ?                        │        263,168 │
│ (MultiHeadGraphAttention)        │                          │                │
├──────────────────────────────────┼──────────────────────────┼────────────────┤
│ multi_head_graph_attention_11    │ ?                        │        263,168 │
│ (MultiHeadGraphAttention)        │                          │                │
├──────────────────────────────────┼──────────────────────────┼────────────────┤
│ multi_head_graph_attention_12    │ ?                        │        263,168 │
│ (MultiHeadGraphAttention)        │                          │                │
├──────────────────────────────────┼──────────────────────────┼────────────────┤
│ multi_head_graph_attention_13    │ ?                        │        263,168 │
│ (MultiHeadGraphAttention)        │                          │                │
├──────────────────────────────────┼──────────────────────────┼────────────────┤
│ dense_5 (Dense)                  │ (2708, 7)                │          3,591 │
└──────────────────────────────────┴──────────────────────────┴────────────────┘

 Total params: 5,686,288 (21.69 MB)

 Trainable params: 2,843,143 (10.85 MB)

 Non-trainable params: 0 (0.00 B)

 Optimizer params: 2,843,145 (10.85 MB)
```

Figure 6: Model Summary of GAT 1 Architecture

5.5 Graph Attention Networks (GAT) Architecture 2

The final graph-based Node Classification GAT model proposed was built by utilizing the attention mechanisms in which the author has used "MultiHeadGraphAttention" layers which were based on using Attention mechanisms. The author used several hyperparameters such as the number of hidden units as 100, the number of heads as 8, number of hidden layers as 6 and then the model was trained on the data for 50 epochs. This was the whole architecture of our proposed Graph Attention Networks (GAT) Node Classifier which includes 5,002,407 trainable parameters and 0 non-trainable parameters as can be seen in the model summary in Figure 7. In this architecture, the optimizer that was



```
Model: "graph_attention_network_3"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━┓
┃ Layer (type)                      ┃ Output Shape           ┃      Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━┩
│ dense_6 (Dense)                   │ (2708, 800)            │    1,147,200 │
├───────────────────────────────────┼────────────────────────┼──────────────┤
│ multi_head_graph_attention_14     │ ?                      │      641,600 │
│ (MultiHeadGraphAttention)         │                        │              │
├───────────────────────────────────┼────────────────────────┼──────────────┤
│ multi_head_graph_attention_15     │ ?                      │      641,600 │
│ (MultiHeadGraphAttention)         │                        │              │
├───────────────────────────────────┼────────────────────────┼──────────────┤
│ multi_head_graph_attention_16     │ ?                      │      641,600 │
│ (MultiHeadGraphAttention)         │                        │              │
├───────────────────────────────────┼────────────────────────┼──────────────┤
│ multi_head_graph_attention_17     │ ?                      │      641,600 │
│ (MultiHeadGraphAttention)         │                        │              │
├───────────────────────────────────┼────────────────────────┼──────────────┤
│ multi_head_graph_attention_18     │ ?                      │      641,600 │
│ (MultiHeadGraphAttention)         │                        │              │
├───────────────────────────────────┼────────────────────────┼──────────────┤
│ multi_head_graph_attention_19     │ ?                      │      641,600 │
│ (MultiHeadGraphAttention)         │                        │              │
├───────────────────────────────────┼────────────────────────┼──────────────┤
│ dense_7 (Dense)                   │ (2708, 7)              │        5,607 │
└───────────────────────────────────┴────────────────────────┴──────────────┘

 Total params: 10,004,816 (38.17 MB)

 Trainable params: 5,002,407 (19.08 MB)

 Non-trainable params: 0 (0.00 B)

 Optimizer params: 5,002,409 (19.08 MB)
```

Figure 7: Model Summary of GAT 2 Architecture

utilized was Stochastic Gradient Descent (SGD) with a momentum of 0.9 and a learning rate of 0.01.

6 Evaluation

In this section, the author has discussed the evaluation results of Keras baseline architectures as well as the different proposed architectures which include Graph Neural Network (GNN) Node classifier and the graph-based models based on attention mechanisms such as Graph Attention Networks which were better from GNN models as it uses attention mechanisms for node classification and classification tasks. In this study, the author has used validation accuracy and validation losses as the evaluation metrics for assessing the performance of the models and finally done the prediction of the randomly generated instances. Search outputs and levels of significance. Experiments 1-2 cover Research Objective 2, while Experiments 3-4 cover Research Objective 3.

6.1 Baseline Architecture

In the study, the baseline GNN Node classifier which can be considered as the most basic architecture of graph neural networks-based Node classifier without tuning the hyperparameters was trained on the dataset of CORA research papers and CORA citations. Initially, when the model was compiled and trained for 300 epochs, the model was automatically stopped by specified callbacks i.e., due to early stopping which was specified while fitting the training data on this baseline architecture. The model was stopped after the 62nd epoch as it was prone to overfitting and performing well on the training data but not on the testing data and got a validation accuracy of 70.71% only also it can be that after reaching a certain point the validation loss is not decreasing as well as it can be seen in the Figure 8. The prediction of the randomly generated instances was also done using this baseline architecture.

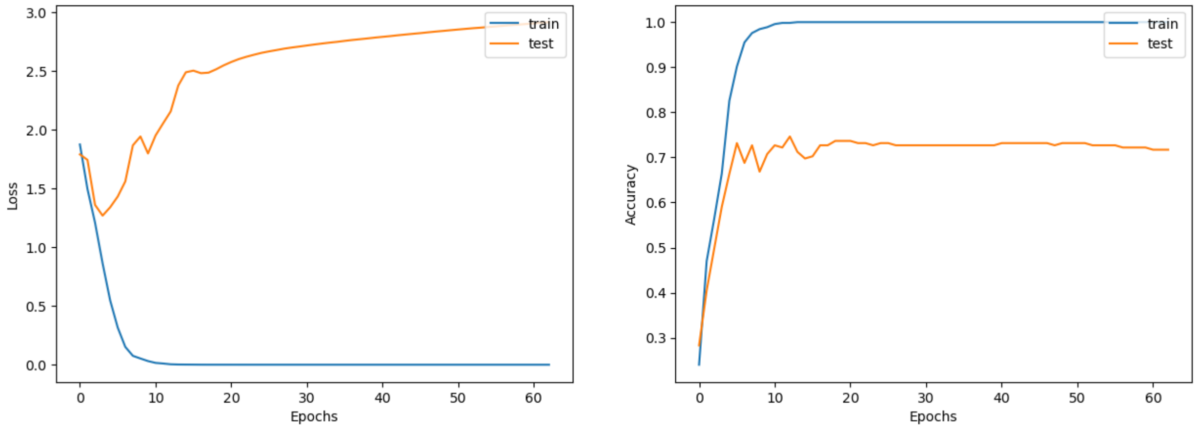


Figure 8: Validation Losses and Accuracy Plots of Baseline GNN Node Classifier

6.2 Experiment 1: Graph Neural Network (GNN) Node Classifier Architecture 1

As the author has seen the baseline architecture was not giving good accuracy, the author has tried and tested a similar architecture of GNN Node Classifier with different hyperparameters in the author's proposed architecture of GNN Node classifier has used 8 Graph Convolutional layers with 100 neurons in each layer with a dropout rate of 0.5 and trained it for 150 epochs, and have also noticed that at certain epoch the validation accuracy was not increasing and due to early stopping this model's was stopped and evaluated based on its validation accuracy which was 69.96% after 63rd epoch and as it can be seen in Figure 9. After applying some changes to the baseline architecture, the author proposed a not slightly better model than the baseline ones and finally the prediction was done using this proposed GNN Node classifier model as well on the same randomly generated instances.

6.3 Experiment 2: Graph Neural Network (GNN) Node Classifier Architecture 2 (Tuned Hyperparameters)

As the author has got a better validation accuracy on the first proposed Graph Neural Networks-based architecture as compared to the baseline architecture, so the author has tried with another experiment of this study such that doing hyperparameter tuning using Keras tuner of the previously proposed architecture in Experiment 1 by tuning the number of hidden units, rate of dropout, type of aggregation, and type of combination in the GNN based Node classifier and tuner was ran for 100 epochs each trial for maximum 15 trials and found the best combination of the hyperparameters i.e., optimal number of hidden units as [88,96], dropout rate as 0.4, aggregation type is the sum, combination type is

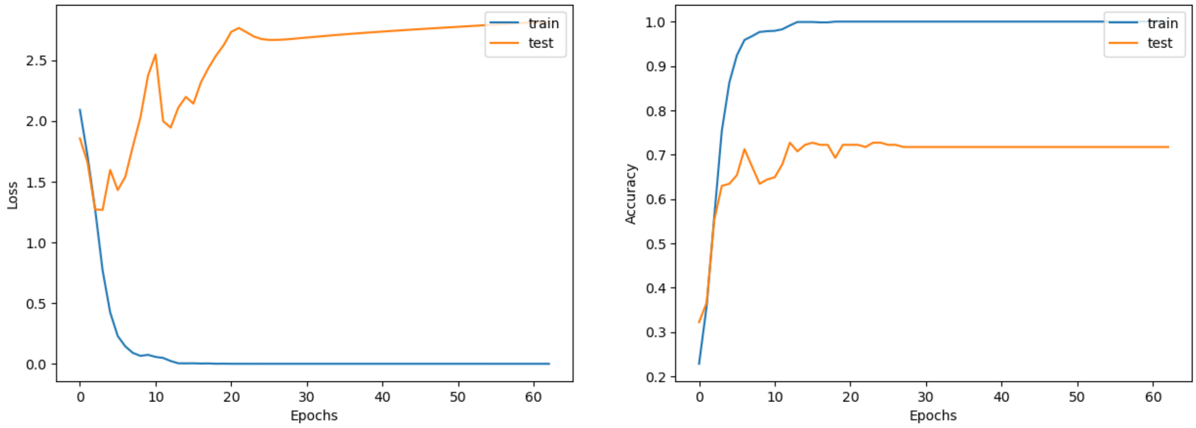


Figure 9: Validation Losses and Accuracy Plots of Proposed GNN Architecture 1

added and normalization setting as False. With these hyperparameters, the proposed GNN Node Classifier was built and trained for three hundred epochs but this was also stopped after the 62nd epoch and got an accuracy of 71.18% which was better than the first proposed GNN Node classifier and the baseline architecture as we can see the plots in Figure 10 In this proposed GNN architecture 2, the author has optimised some

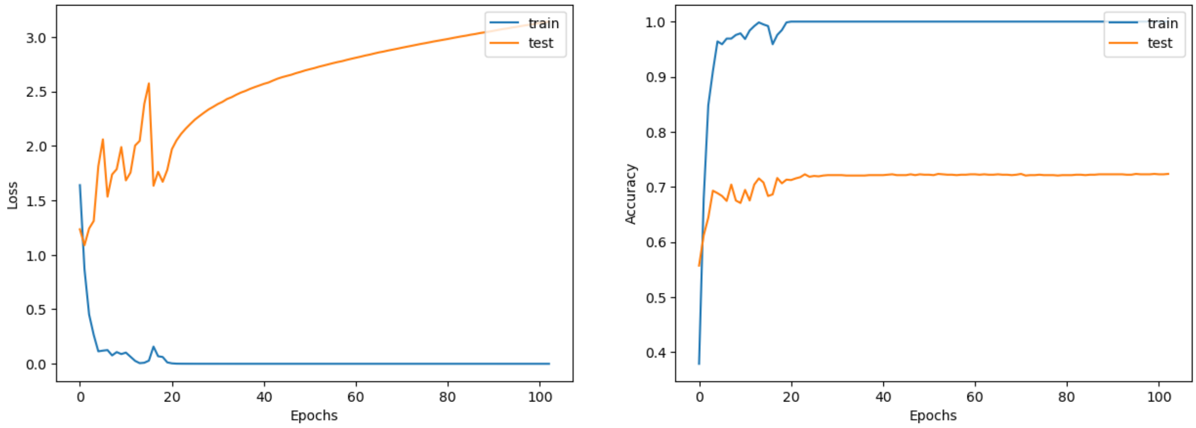


Figure 10: Validation Losses and Accuracy Plots of Proposed GNN Architecture 2

significant hyperparameters such as the number of hidden units, dropout rate, type of aggregation, combination, and normalization settings. However it has been observed that there was not much improvement from the baseline GNN architecture and the proposed GNN architecture i.e., experiment 1, there could be some of these possible reasons such as the proposed model's hyperparameter search space being limited as we can see in the code[3] that search space for the number of hidden units was considered with min_value of 8 and the max_value of 128 with a step size of 8, the dropout rate was considered with the min_value of 0.0 to the max_value of 0.5 with a step 0.1. The other reason could be that the proposed GNN architecture in the experiment has reached a certain point of convergence where the search space of the hyperparameters could only result in very

[3] https://github.com/yashbhargava16/Graph-Neural-Networks

little improvement in the results. Most importantly, the dataset used for this proposed model i.e., the CORA dataset was small which could have limited the model's ability to generalize the optimized hyperparameters which shows that not much improvement in the results from the baseline GNN architecture.

6.4 Experiment 3: Graph Attention Networks (GAT) Architecture 1

As the author has experimented on two GNN Node classifier architectures and tried to improve the performance of the baseline architectures the author has proposed another graph neural network which was based on attention mechanisms known as Graph Attention Networks and the architecture was built using a certain number of hyperparameters such as number of hidden units, number of heads, number of layers, batch size and learning rate and the architecture was proposed and trained for 50 epochs and stopped at 27th epoch and achieving an overall accuracy of 72.3% which was better than the baseline architecture and the above proposed GNN Node Classifiers models. Also, in these models, the overfitting of the above two proposed architectures has been reduced and the performance of these models has been increased by using the attention mechanisms for graph neural networks and the node classification tasks as the plots of this model can be seen in Figure 11.

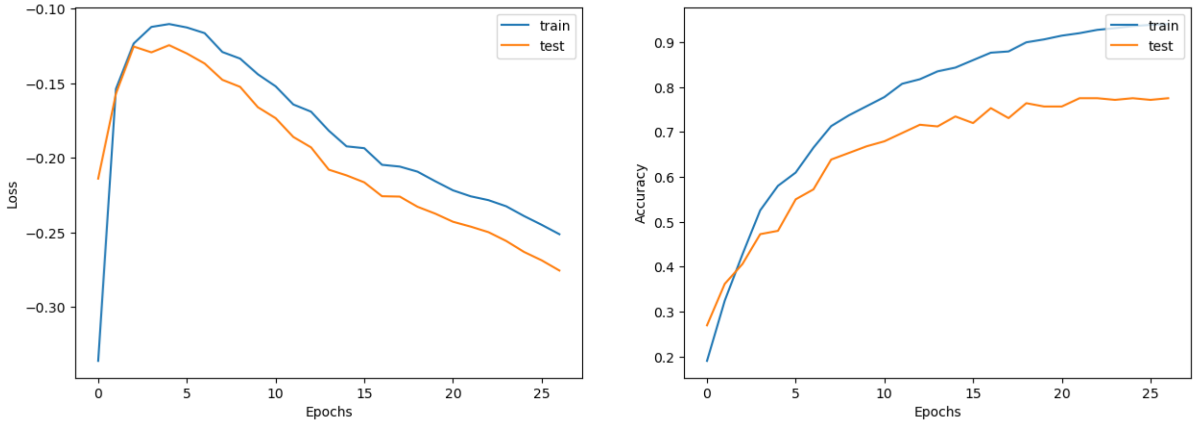


Figure 11: Validation Losses and Accuracy Plots of Proposed GAT Architecture 3

6.5 Experiment 4: Graph Attention Networks (GAT) Architecture 2

Based on attention mechanisms for graph-based neural networks, this study proposed another architecture based on the previous experiment. Still, there were certain changes with the number of hidden units and number of layers in this architecture the author has changed the number of hidden units to 100 and the number of layers to 6 and was trained this proposed model for the same fifty epochs. Now the early stopping of this architecture has happened at the 32nd epoch itself. The model achieved an overall validation accuracy of 73.8% which was slightly lesser than the previous GAT architecture which was best among all the GNN-based Node classifiers and proposed graph attention network model as well in experiment 3 which ensures that more accurate Node classification and plots of this proposed model can be seen in Figure 12.

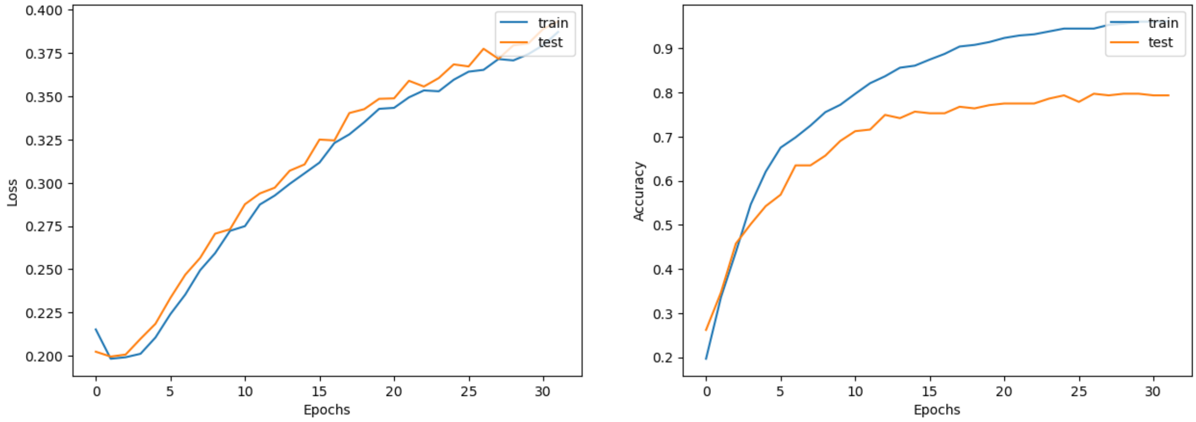


Figure 12: Validation Losses and Accuracy Plots of Proposed GAT Architecture 2

As the code given in the study [4] for the baseline classifier in a simple feed-forward network which was used in preprocessing and postprocessing in GNNs and they have considered this as the baseline architecture. This study has considered the baseline architecture as the baseline GNN architecture provided by Keras [5]. The difference in the plots between the submitted code[6] and the report can be due to several reasons such as the splitting of data as in this case the data was split into train and test by taking the random samples in both the sets due to if the model was retrained each time, the plots will be different. While it is unlikely to be a coding error, minor changes in the implementation could be causing the observed differences.

GNN Models	Validation Accuracy
Baseline Architecture	70.71%
GNN Architecture 1	69.96%
GNN Architecture 2	71.18%
GAT Architecture 1	72.3%
GAT Architecture 2	73.8%

Table 1: GNN architectures with their validation accuracies

These were some architectural changes experimented with to find the optimised balance between the complexity of the model and generalizability in which GAT models outperformed the proposed GNN architectures and the baseline ones.

6.6 Discussion

Validation Accuracy and Validation losses

Several numbers of experiments have been done and various architectures and tuning parameters have been explored so far in graph-based neural networks. In this study, the

[4] https://keras.io/examples/graph/gnn_citations/

[5] https://keras.io/examples/graph/gnn_citations/

[6] https://github.com/yashbhargava16/Graph-Neural-Networks

author has proposed various architectures of Graph Neural Networks (GNNs) and Graph Attention Networks (GATs) with which the author tried to propose those architectures which were better than the baseline architecture in terms of model complexity, validation accuracy, and the efficiency of this model. However, based on the experiments done by the author, one of the key findings of this study was that increasing the complexity of the model such as increasing the number of hidden units, and number of layers does not always improve the performance of the graph neural network (GNN) model. Suppose as it can be seen experimenting with a greater number of Graph Convolutional layers in GNN Architecture 1 has not improved the model's validation accuracy but did not improve as we expected which shows that only adding a greater number of layers and a greater number of hidden units might not be an optimized way to achieve good results, but it might increase the complexity of the model and causes overfitting as well.
In terms of validation accuracy, the different architectures have different validation accuracy as it can seen that the baseline architecture has a validation accuracy of 70.71% which shows that it is not able to capture the relations between the nodes. The proposed GNN Node classifier architecture 1 does not show an improvement in its validation accuracy of 69.96% by tuning and adding more layers. However, experiment 4 where the Graph Attention Networks (GAT) Architecture 2 was proposed has improved its validation accuracy and performed the best with a validation accuracy of 73.8%. This means that utilizing attention mechanisms for graph neural networks can significantly increase the model‘s performance for the tasks of node classification where the relationship between the nodes is complex.
The interpretation of the validation accuracies and validation losses of the plotted curves of those proposed GNN and GAT architectures for these validation accuracies curves for each experiment if the training accuracies of those models were continuously increasing and the validation accuracies remain almost the same after every epoch or if it starts decreasing it an indication of overfitting, similarly for the validation losses curves if the validation loss starts increasing after a certain point of time and training loss was continuously decreasing, it shows that the model is overfitting as the model is not able to generalize on the testing data.In Experiments 1 and 2, the proposed architectures, overfitting can be seen as in the plotted curves of the GNNs the validation loss for the test set was increasing and the validation accuracy remained constant after a certain number of epochs which creates a huge gap between the two curves which indicates the overfitting was happening. Similarly, in the plotted curves for the proposed GATs in experiments 3 and 4 the val_loss continuously decreases with the training loss and validation accuracy is increasing which for the first GAT architecture and validation loss for the second Architecture increased with the training loss simultaneously and the validation accuracy was increasing, which shows very little gap between the two plotted curves which results in less overfitting of those two architectures.

Running time

In terms of the running time of the model, which was training, and the validation time of any model was an important aspect of this research. In this study, the proposed architectures were trained on enough epochs and the author has specified callbacks as an early stopping mechanism to prevent overfitting of the models. The baseline architecture and the other two proposed GNN Node classifiers were stopped around the 60th epoch and 100th epoch, because of overfitting. However, if the proposed Graph attention Networks

(GAT) model which achieved the highest accuracy of 73.8% has only run till the 32nd epoch as it has converged faster than the proposed GNNs which means that utilization of graph attention-based models provides better performance and the cost of computation. The findings of this study align with previous research works that have been discussed in Section 2, and show the potential of Graph Attention Networks (GATs) over the other proposed Graph Neural Networks (GNNs). The improved performance of the GAT model in Experiment 3 is consistent with the related work, which shows that attention mechanisms better capture the relationship between entities in a graph.

6.7 Limitations

This study provides valuable findings in Graph Neural Networks and Graph Attention Networks. However, the author cannot claim to have exhausted all possible neural architectures for GNN or GAT. Another obvious limitation is the focus on only one dataset. The neural network architecture might not apply to other domains. The work does show that the attention mechanism is relevant for the CORA dataset, but this finding might not be generalizable to different domains.

Another limitation of this study was only relying on early stopping to prevent overfitting of the proposed models which does not allow some of the models to utilize their maximum potential. Another limitation was the computational resources available for this study. The complexity and depth of the models were limited by hardware limitations, which does not allow this study to explore more advanced architectures, such as deeper GNNs or more complex GATs. These are some of the limitations of this study that could be an interesting starting point for future investigations.

7 Conclusion and Future Work

Graph neural networks and attention mechanisms are well-known techniques for the classification of graph data. Little is known about suitable neural architectures for a given problem. In this work, a range of GNN and GAT architectures were proposed and compared for node classification. To do so, this study built a graph neural network-based model with improved validation accuracy for an existing baseline GNN provided by Keras.The author has proposed two GNN Node classifier models and two Graph Attention network models and tuned the hyperparameters of the proposed model by experimenting with different architectures to evaluate the models using validation accuracy and losses. Learnings from this research can be summarized as follows:

- Improved the classification accuracy of nodes by utilizing Graph Attention Networks (GAT) based on attention mechanisms. The model outperformed any proposed GNN Node classifier models and the baseline model architecture as well by achieving the validation accuracy of 73.8%.

- Adding more layers and units does not always show improvement in the performance of the model but increases the complexity of the GNN-based models.

- Overall, this study showed that utilizing attention mechanisms in Graph neural network-based models improved the performance dramatically as compared to the simpler GNN-based model.

All in all, the work answered the research question. Graph Attention Network (GAT) architectures outperform GNNs for node classification (CORA dataset). However, there were some limitations as well such as the study focused on only one dataset, Interestingly, only relying on early stopping was not a good strategy to avoid overfitting and the size of the dataset was relatively small. In other words, the proposed architecture might not be suitable as a general-purpose GAT network, but it could be seen as a starting architecture for further tuning.

In future work, this study can be expanded by addressing some of the limitations of the study such as using more different datasets other than the CORA dataset which can help improve the generalizability of the proposed architectures in this study, which allows these models to become more robust and overcome the current limitations of this study which has used only one dataset. As the edge weights were normalised and hard wired into the model which results the model to behave statically and utilizes the original edge weights which were assigned while training the model. When the new articles will be added to the graph, instead of doing the retraining of the model, in future various approaches can be considered like making the computation of the edge weights dynamic instead of doing it hardly coded and transfer learning can also be utilized by fine tuning the pre-trained proposed GNN architectures which can be helpful in enhancing the classification performance as well. Several other techniques to prevent the model from overfitting can be employed for further research, such as regularization techniques and combinational models. Finally, there is a possibility for using these findings, especially in industries that are dependent on complex data structures, such as finance, healthcare, and social network analysis.

References

Bera, A., Bhattacharjee, D. and Krejcar, O. (2024). Pnd-net: plant nutrition deficiency and disease classification using graph convolutional network, *Scientific Reports* **14**(1): 15537.

Besta, M., Catarino, A. C., Gianinazzi, L., Blach, N., Nyczyk, P., Niewiadomski, H. and Hoefler, T. (2024). Hot: Higher-order dynamic graph representation learning with efficient transformers, *Learning on Graphs Conference*, PMLR, pp. 15–1.

Chen, Y., Xiao, Z., Du, Y., Zhao, L., Zhang, L., Wu, Z., Zhu, D., Zhang, T., Yao, D., Hu, X. et al. (2024). A unified and biologically plausible relational graph representation of vision transformers, *IEEE Transactions on Neural Networks and Learning Systems* .

Chen, Y., Zhang, F., Wang, M., Zekelman, L. R., Cetin-Karayumak, S., Xue, T., Zhang, C., Song, Y., Makris, N., Rathi, Y. et al. (2024). Tractgraphformer: Anatomically informed hybrid graph cnn-transformer network for classification from diffusion mri tractography, *arXiv preprint arXiv:2407.08883* .

Chung, J., Ahmad, R., Sun, W., Cai, W. and Mukerji, T. (2024). Prediction of effective elastic moduli of rocks using graph neural networks, *Computer Methods in Applied Mechanics and Engineering* **421**: 116780.

Dash, B. M., Bouamama, B. O., Boukerdja, M. and Pekpe, K. M. (2024). Bond graph-cnn based hybrid fault diagnosis with minimum labeled data, *Engineering Applications of Artificial Intelligence* **131**: 107734.

El-Gayar, M. M., Abouhawwash, M., Askar, S. S. and Sweidan, S. (2024). A novel approach for detecting deep fake videos using graph neural network, *Journal of Big Data* **11**(1): 22.

Gao, M., Zhang, D., Chen, Y., Zhang, Y., Wang, Z., Wang, X., Li, S., Guo, Y., Webb, G. I., Nguyen, A. T. et al. (2024). Graphormerdti: a graph transformer-based approach for drug-target interaction prediction, *Computers in Biology and Medicine* **173**: 108339.

Jin, Y. (2024). Graphcnnpred: A stock market indices prediction using a graph based deep learning system, *arXiv preprint arXiv:2407.03760* .

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks, *arXiv preprint arXiv:1609.02907* .

Kofinas, M., Knyazev, B., Zhang, Y., Chen, Y., Burghouts, G. J., Gavves, E., Snoek, C. G. and Zhang, D. W. (2024). Graph neural networks for learning equivariant representations of neural networks, *arXiv preprint arXiv:2403.12143* .

Kouris, P., Alexandridis, G. and Stafylopatis, A. (2024). Text summarization based on semantic graphs: An abstract meaning representation graph-to-text deep learning approach, *Journal of Big Data* **11**(1): 95.

Li, G., Muller, M., Thabet, A. and Ghanem, B. (2019). Deepgcns: Can gcns go as deep as cnns?, *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9267–9276.

Lu, Y., Li, Y., Zhang, R., Chen, W., Ai, B. and Niyato, D. (2024). Graph neural networks for wireless networks: Graph representation, architecture and evaluation, *arXiv preprint arXiv:2404.11858* .

Monroy, L. C. R., Rist, L., Wilm, F., Ostalecki, C., Baur, A., Vera, J., Breininger, K. and Maier, A. (2024). Multi-level cancer profiling through joint cell-graph representations, *Smart Health* **32**: 100470.

Ning, D., Vetrova, V., Bryan, K. R. and Koh, Y. S. (2024). Harnessing the power of graph representation in climate forecasting: Predicting global monthly mean sea surface temperatures and anomalies, *Earth and Space Science* **11**(3): e2023EA003455.

Ramirez, R., Chiu, Y.-C., Hererra, A., Mostavi, M., Ramirez, J., Chen, Y., Huang, Y. and Jin, Y.-F. (2020). Classification of cancer types using graph convolutional neural networks, *Frontiers in physics* **8**: 203.

Shi, J., Chaudhari, S. and Moura, J. M. (2024). Graph convolutional neural networks in the companion model, *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp. 7045–7049.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. and Philip, S. Y. (2020). A comprehensive survey on graph neural networks, *IEEE transactions on neural networks and learning systems* **32**(1): 4–24.