NODE CLASSIFICATION ON THE CITATION NETWORK USING GRAPH NEURAL NETWORK

Irani Hoeronis[1*], Bambang Riyanto Trilaksono[2]
Informatics, Siliwangi University, Tasikmalaya, Indonesia[1]
School of Electrical and Informatics Engineering, Bandung Institute of Technology, Bandung, Indonesia[2]
E-mail address: iranihoeronis@unsil.ac.id[1], briyanto@lskk.ee.itb.ac.id[2]

Received: 7, June, 2023 ***Revised: 27, June, 2023*** ***Accepted: 30, June, 2023***

ABSTRACT

Research on Graph Neural Networks has influenced various current real-world problems. The graph-based approach is considered capable of effectively representing the actual state of surrounding data by utilizing nodes, edges, and features. Consider the feedforward neural network and the graph neural network approaches, we determine the accuracy of each method. In the baseline experiment, training and testing were performed using the NN approach. The resulting accuracy of FNN was 72.59% and GNN model has increased by 81.65%. There is a 9.06% increase in accuracy between the baseline model and the GNN model. The new data utilized in the model predictions showcases the probabilities of each class through randomly generated examples.

Keywords: Graph Neural Network, FeedForward Neural Network, Node Classification

1. INTRODUCTION

The real-world representation is currently growing rapidly. Several approaches are being taken to solve problems in the real world. The closer the model's representation is to real conditions, the more accurate the predictions and the smaller the resulting error. The variety of data in the real world necessitates the adjustment of the model's representation of the data. Initially, the data could only be addressed in a linear form. Each data point in real-world problems was transformed into a linear format, and subsequently, each formed model yielded a specific value based on its linearity. Numerous non-linear approaches have emerged for solving real-world problems. This pertains to data that exhibits an irregular structure, such as maps connecting cities in a single location, citation networks with distinct features on each node of an edge, 3D graph shapes, molecular structures, social networks, and so forth. Deep learning methods enable computers to learn from various types of data, including information about people, animals, algorithms, or agents from general-purpose computers.

The learning process is employed to enhance future performance. In other words, the learning process converts experience into expertise or knowledge. The learning algorithm takes training data as input, which represents experience, and produces expertise as output in the form of computer programs, predictive models, or fine-tuning of internal variables. The definition of performance depends on the specific algorithm or the desired goal. The objective

is to generate predictions tailored to the needs that are influenced by the data's condition. The data's quality significantly impacts performance, thus ensuring that the data used aligns with the problem-solving focus is crucial.

The deep learning process is conducted with two main components: learning algorithms and data (Negro, 2020). The choice of learning algorithm in deep learning depends on the specific problem at hand. As for the data used, it encompasses a wide range, from matrices and tensors to sequences and time series (Ma & Tang, 2020). The diversity of real-world data presents challenges in terms of formulation due to its irregular shape. Data exhibits complex relational structures that need to be extracted to understand how objects interact with one another. Graphs serve as a universal data structure capable of representing intricate data relationships, and they find applications in multiple domains such as computational chemistry, social networks (Bai et al., 2021), recommendation systems, biology, and more.

Various data types can be transformed into graphs (Xu, 2017). Furthermore, the abundance of values in real-world problems can be processed as a set of computations that are more efficiently handled on graphs. Presently, graphs are recognized as powerful tools for problem formulation. They serve as the data to be processed by learning algorithms to achieve improved performance. Graph Neural Networks conduct analysis at the level of nodes, edges, or the entire graph.

At the node level, connected data graphs are considered independent, unlike traditional deep learning techniques that assume data to be dependent and identically distributed, which is not compatible with graph computation. Graph analysis focusing on node classification (Arul Prakash & Tucker, 2021)(Yao et al., 2022), link prediction, and clustering has been extensively conducted (Li et al., 2020)(Tsitsulin et al., 2020)(Zhang et al., 2022). Furthermore, classification and clustering are also performed at the graph level through semi-supervised and self-supervised learning approaches (Liu et al., 2022)(Kipf & Welling, 2017)(R. Guo et al., 2022). These factors serve as motivation for the research and development of graph-based predictive models.

The application and implementation of Graph Neural Networks are widely diverse, spanning various disciplines. Some notable applications include Personal Assistants, Computer Vision, Recommender systems, predictions in the health sector (Lin et al., 2023)(S. Guo et al., 2021)(Weng & Zhu, 2021), biology (bioinformatics), chemistry, and numerous others. In this study, the graph neural network approach is utilized for node classification. Apart from generating accuracy values to assess the model's predictive performance, node embedding is also employed to classify class categories based on citation data within the network.

2. THEORY

Research on Graph Neural Networks encompasses various implementations. This paper aims to describe the fundamental theory of graph neural networks and explore their scope.

DEEP LEARNING

Known also as deep neural networks, artificial neural networks with numerous layers are the subject of deep learning, a subfield of machine learning. Deep learning's primary objective is to model and acquire hierarchical data representations so that the networks can automatically

recognize complex patterns and characteristics from raw input. Artificial neural networks, which are computational models inspired by the structure and operation of biological brains, are the fundamental building block upon which deep learning is based. These networks are made up of interconnected, layered nodes known as neurons or units. Each neuron takes in information from the layer above, applies an activation function, and produces an output that becomes the input for the layer above.

A key component of deep learning is the depth of neural networks since it makes it possible to learn hierarchical representations. Deep neural networks gradually learn complicated and abstract interpretations of the data by layering several levels. Each layer takes the necessary features from the incoming data and passes them on to succeeding levels, producing ever more complex and abstract representations. Deep learning's training procedure mainly on the backpropagation algorithm. The network modifies its internal parameters, or weights, throughout training in response to differences between expected and actual results. The network learns to reduce errors and improve performance on the given job through iterative weight modifications.

Convolutional neural networks (CNNs) for image and video processing represent a significant improvement in deep learning. For tasks like image classification, object detection, and picture segmentation, CNNs use convolutional layers that effectively capture spatial relationships within the data. Recurrent neural networks (RNNs) are another significant advancement in deep learning that are used to interpret sequential and temporal input. Text, speech, and time series data, as well as data with sequential dependencies, can all be handled by RNNs. Recurrent connections enable RNNs to maintain internal memory and analyze data sequentially, making them useful for tasks like machine translation, speech recognition, and natural language processing.

Deep learning has excelled in several fields, including reinforcement learning, computer vision, natural language processing, and speech recognition. The development of artificial intelligence has been spurred by its capacity to automatically acquire hierarchical representations from raw data in a variety of sectors, including image recognition, autonomous driving, drug discovery, and many more.

GRAPH NEURAL NETWORK

The data is represented in the form of a graph, comprising nodes, edges, and features. A graph is denoted as G=(V, E, u), where G represents the graph, V represents the vertices (nodes), E represents the edges (adjacency, weight), and u represents the feature vector. Graph data has a complexity that is formed from the many challenges of deep learning algorithms. The reason is that conventional deep learning and deep learning tools specialize in simple data types. For example, an image object with the same size and structure, where the image is a fixed-size grid graph. Just as text and speech are sequential, we can think of them as line graphs.

Unlike the case with complex graphs, without a fixed shape with variable sizes of undefined nodes, where nodes have a different number of neighbor nodes. This also has different assumptions from the existing deep learning algorithms. The current DL algorithm

assumes that each instance is independent from one another. But it is different from graph data because each node is connected to other nodes with a variety of link types. The process of designing a Graph Neural Network model includes the input process in the form of a graph. In processing the graph into the GNN layer, it is necessary to determine the structure of the graph, as well as the type and size of the graph.

Furthermore, the model is constructed computationally utilizing a variety of GNN layer methods. Embeddings on nodes, edges, or graphs are the result of this process. Designing a loss function to minimize its value is the last step. The type of training and the task at hand are considered while determining the loss function. Unsupervised learning, semi-supervised learning, and supervised learning are all possible training methods. At the node, edge, or graph levels, the task level can be carried out.

3. METHOD

To compare the methods used in node classification, we consider the feedforward neural network and the graph neural network approaches. Subsequently, we determine the accuracy of each method.

FEED FORWARD NETWORK

Feedforward neural networks have been used in a variety of machine learning applications, including as classification, regression, and pattern recognition. They are capable of learning complex nonlinear correlations in the data. Three main parts make up the construction of a feedforward neural network: an input layer, one or more hidden layers, and an output layer. Multiple interconnected artificial neurons, often referred to as nodes or units, make up each layer. With connections between the nodes in adjacent levels, these neurons are arranged in a hierarchy of layers.

Without any feedback connections, information moves forward through the layers of a feedforward neural network. Each neuron in a layer requires input from the neurons in the layer further it, performs a weighted sum of these inputs, applies an activation function to the sum, and then outputs the result to the neurons in the layer above. Up until the output layer, which results in the network's ultimate output, this process is repeated for every layer.

The factors that determine the network's behavior are the weights and biases connected to its connections between neurons. Using optimization algorithms like gradient descent, these parameters are changed during the training phase with the aim of reducing a specified loss function that gauges the disparity between the anticipated output and the desired result. A total of 62,507 parameters were used in the feedforward neural network. The constructed network structure comprises 2 feedforward neural network (FFN) blocks, 1 skip connection, 3 repetitions of 1 FFN block and 1 skip connection, and 1 dense layer. Table 1 depicts the structure of the feedforward neural network (FFN) model.

Table 1. Structure of Feedforward Neural Network (FFN) Model

Layer (type)	Output Shape	Param #	Connected to
input_features (InputLayer)	[(None, 1433)]	0	[]
ffn_block1 (Sequential)	(None, 32)	52804	[‘input_features[0][0]’]
ffn_block2 (Sequential)	(None, 32)	2368	[‘ffn_block1[0][0]’]
skip_connection2 (Add)	(None, 32)	0	[‘ffn_block1[0][0]’, ‘ffn_block2[0][0]’]
ffn_block3 (Sequential)	(None, 32)	2368	[‘skip_connection2[0][0]’]
skip_connection3 (Add)	(None, 32)	0	[‘skip_connection2[0][0]’, ‘ffn_block3[0][0]’]
ffn_block4 (Sequential)	(None, 32)	2368	[‘skip_connection3[0][0]’]
skip_connection4 (Add)	(None, 32)	0	[‘skip_connection3[0][0]’, ‘ffn_block4[0][0]’]
ffn_block5 (Sequential)	(None, 32)	2368	[‘skip_connection4[0][0]’]
skip_connection5 (Add)	(None, 32)	0	[‘skip_connection4[0][0]’, ‘ffn_block5[0][0]’]
Logits (Dense)	(None, 7)	231	[‘skip_connection5[0][0]’]

GRAPH NEURAL NETWORK

The key idea behind GNNs is to propagate and aggregate information through the nodes and edges of a graph to capture and model the underlying relationships and structural properties of the graph. GNNs typically operate in multiple iterations, known as message passing or graph convolutional layers, where information is exchanged between connected nodes and aggregated to update the node representations. At each layer, the GNN processes the node features and the features of neighboring nodes to generate updated node representations. These updated representations incorporate information from the local neighborhood of each node, allowing the GNN to capture both the local and global structural patterns of the graph.

The aggregation step typically involves aggregating the information from neighboring nodes using some form of pooling. A total of 67,179 parameters were used in the graph neural network. The constructed network structure comprises preprocess, 2 graph convolution layer, proprocess, and dense layer. Table 2 depicts the structure of the graph neural network (GNN) model.

Table 2. Structure of Graph Neural Network (GNN) Model

Layer (type)	Output Shape	Param #
preprocess (Sequential)	(2708, 32)	52804
graph_conv1 (GraphConvLayer)	multiple	5888
graph_conv1 (GraphConvLayer)	multiple	5888
postprocess (Sequential)	(2708, 32)	2368
logit (Dense)	multiple	231

4. RESULTS AND DISCUSSION

Research on graph neural networks is broadly categorized into three levels: node, edge, and graph levels. The initial level of research focuses on node classification. At the edge level, research is conducted on link prediction, utilizing the information from interconnected nodes. Subsequent research progresses to the graph level, where node and edge information is leveraged to obtain a new representation of the entire data graph. To extract graph datasets, the system was constructed using the Python programming language along with the Keras and networkx libraries. The model architecture consists of 2 convolution layers. The Cora dataset is employed to predict the subject of scientific papers and analyze the citation network.

The Cora dataset encompasses 2,708 scientific papers, categorized into 7 classes, with a network of 5,429 citation links. Each paper is associated with a binary word vector of size 1,433, indicating the presence of specific words. The subjects consists of 7 categories, which are Case_Based, Genetic_Algorithms, Neural_Networks, Probabilistic_Methods, Reinforcement_Learning, Rule_Learning, and Theory. The dataset is divided into two separate files, namely cora.cites and cora.content. cora.cites contains citation records with two columns: cited_paper_id (target) and citing_paper_id (source). Figure 1 illustrates the data graph of the Cora dataset.

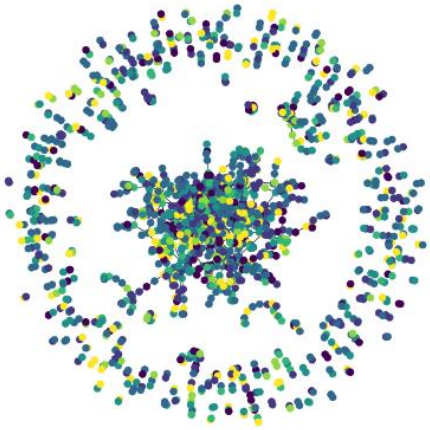

Figure 1. Cora Dataset

Experiments were conducted using a baseline neural network, which was then compared with a graph neural network model. In the baseline experiment, training and testing were performed using the NN approach. A FeedForward Network was added to the NN model with a skip connection. The total number of parameters used for the baseline NN and GNN models is 62,507, with 29,065 trainable parameters and 3,442 non-trainable parameters. The dropout rate utilized was 0.5, the learning rate was set to 0.01, the epoch count was 300, and the batch size was 256. In the split data test, the resulting accuracy was 72.59%. The data employed in the prediction model displays the probability of each class using randomly generated examples. Following the probabilities of each class in the random sample using FNN Approach.

1/1 [[==========] – 0s 287ms/step
Instance 1:
- Case_Based: 37.04%
- Genetic_Algorithms: 20.5%

- Neural_Networks: 19.41%
- Probabilistic_Methods: 10.86%
- Reinforcement_Learning: 3.41%
- Rule_Learning: 6.3%
- Theory: 2.49%

Instance 2:

- Case_Based: 1.11%
- Genetic_Algorithms: 2.51%
- Neural_Networks: 86.6%
- Probabilistic_Methods: 8.42%
- Reinforcement_Learning: 0.45%
- Rule_Learning: 0.56%
- Theory: 0.36%

Instance 3:

- Case_Based: 0.04%
- Genetic_Algorithms: 99.17%
- Neural_Networks: 0.4%
- Probabilistic_Methods: 0.03%
- Reinforcement_Learning: 0.26%
- Rule_Learning: 0.04%
- Theory: 0.06%

The loss and accuray results for the baseline Neural Network are shown in Figure 2.

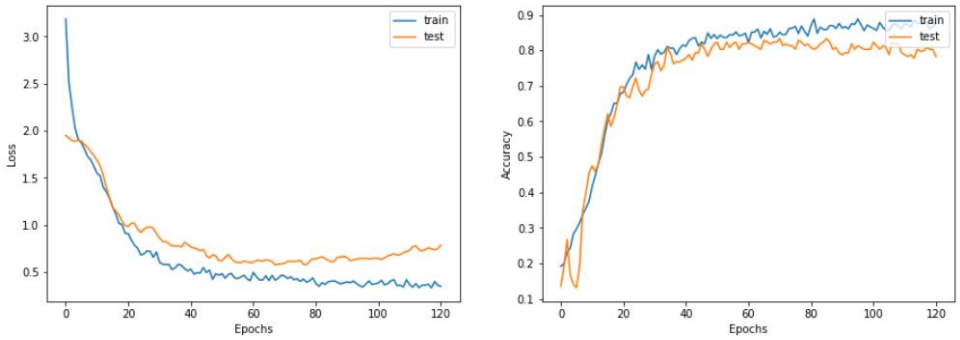

Figure 2 Loss and Accuracy Results of The Baseline Neural Network

The process of building a graph neural network model begins with data graph preparation and its integration into the model for training. This particular aspect of the GNN model utilizing specific libraries presents significant challenges. The numpy library is employed for processing node_features, edges, and edge_weights to represent the graph. In this context, nodes represent papers while edges depict the citations between papers. Notably, the data lacks weights for paper citations. For implementing the graph convolution, the keras layer serves as a module. The process involves three steps: preparation, aggregation, and updating. The GNN functions as a feature extractor and node classifier, with each convolutional graph layer incorporating information from neighboring nodes. The loss and accuracy results of the GNN model are depicted in Figure 3.

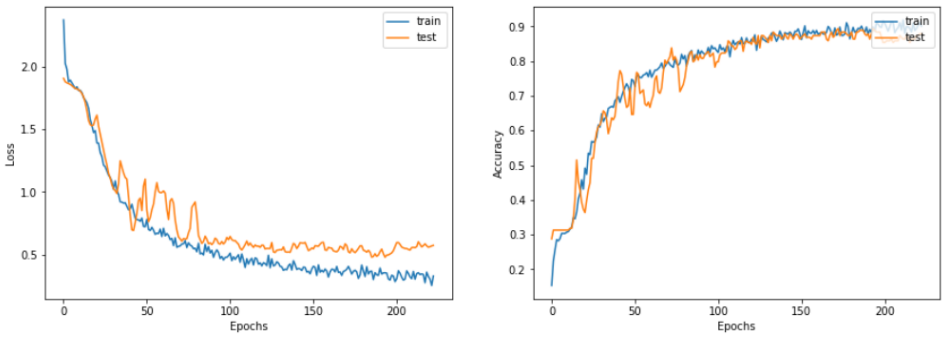

Figure 3 Loss and Accuracy Results in The GNN Model

The accuracy of the GNN model has increased by 81.65%. The new data utilized in the model predictions showcases the probabilities of each class through randomly generated examples. Following the probabilities of each class in the random sample using GNN Model.

1/1 [[==========] – 1s 1ms/step

Instance 1:

- Case_Based: 1.32%
- Genetic_Algorithms: 94.95%
- Neural_Networks: 0.35%
- Probabilistic_Methods: 0.2%
- Reinforcement_Learning: 3.01%
- Rule_Learning: 0.06%
- Theory: 0.11%

Instance 2:

- Case_Based: 0.56%
- Genetic_Algorithms: 57.36%
- Neural_Networks: 14.54%
- Probabilistic_Methods: 0.92%
- Reinforcement_Learning: 26.17%
- Rule_Learning: 0.13%
- Theory: 0.32%

Instance 3:

- Case_Based: 0.48%
- Genetic_Algorithms: 5.6%
- Neural_Networks: 87.09%
- Probabilistic_Methods: 2.86%
- Reinforcement_Learning: 2.16%
- Rule_Learning: 0.69%
- Theory: 1.12%

There is a 9.06% increase in accuracy between the baseline model and the GNN model.

5. CONCLUSIONS AND SUGGESTIONS

Research on Graph Neural Networks has influenced various current real-world problems. The graph-based approach is considered capable of effectively representing the actual state of surrounding data by utilizing nodes, edges, and features. Consider the feedforward neural network and the graph neural network approaches, we determine the accuracy of each method.

In the baseline experiment, training and testing were performed using the NN approach. The resulting accuracy of FNN was 72.59% and GNN model has increased by 81.65%. There is a 9.06% increase in accuracy between the baseline model and the GNN model. The new data utilized in the model predictions showcases the probabilities of each class through randomly generated examples.

For further research, it can be conducted at both the edge and graph levels by predicting journals that share citation keywords with certain journals.

REFERENCES

Arul Prakash, S. K., & Tucker, C. S. (2021). Node classification using kernel propagation in graph neural networks. *Expert Systems with Applications*, *174*(February), 114655. https://doi.org/10.1016/j.eswa.2021.114655

Bai, N., Meng, F., Rui, X., & Wang, Z. (2021). Rumour Detection Based on Graph Convolutional Neural Net. *IEEE Access*, *9*, 21686–21693. https://doi.org/10.1109/ACCESS.2021.3050563

Guo, R., Sun, J., Zhang, C., & Qian, X. (2022). A Self-Supervised Metric Learning Framework for the Arising-From-Chair Assessment of Parkinsonians with Graph Convolutional Networks. *IEEE Transactions on Circuits and Systems for Video Technology*, *32*(9), 6461–6471. https://doi.org/10.1109/TCSVT.2022.3163959

Guo, S., Rigall, E., Qi, L., Dong, X., Li, H., & Dong, J. (2021). Graph-based CNNs with self-supervised module for 3d hand pose estimation from monocular rgb. *IEEE Transactions on Circuits and Systems for Video Technology*, *31*(4), 1514–1525. https://doi.org/10.1109/TCSVT.2020.3004453

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 1–14.

Li, X., Hu, Y., Sun, Y., Hu, J., Zhang, J., & Qu, M. (2020). A Deep Graph Structured Clustering Network. In *IEEE Access* (Vol. 8, pp. 161727–161738). https://doi.org/10.1109/ACCESS.2020.3020192

Lin, L., Xiong, M., Zhang, G., Kang, W., Sun, S., & Wu, S. (2023). A Convolutional Neural Network and Graph Convolutional Network Based Framework for AD Classification. *Sensors*, *23*(4), 3163–3173. https://doi.org/10.3390/s23041914

Liu, Y., Li, Z., Pan, S., Gong, C., Zhou, C., & Karypis, G. (2022). Anomaly Detection on Attributed Networks via Contrastive Self-Supervised Learning. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(6), 2378–2392. https://doi.org/10.1109/TNNLS.2021.3068344

Ma, Y., & Tang, J. (2020). *Deep Learning on Graphs*.

Negro, A. (2020). *Graph Powered Machine Learning*. Manning Publications.

Tsitsulin, A., Palowitch, J., Perozzi, B., & Müller, E. (2020). Graph Clustering with Graph Neural Networks. *Journal of Machine Learning Research*, *24*, 1–21. http://arxiv.org/abs/2006.16904

Weng, W., & Zhu, X. (2021). UNet: Convolutional Networks for Biomedical Image Segmentation. *IEEE Access*, *9*, 16591–16603. https://doi.org/10.1109/ACCESS.2021.3053408

Xu, J. (2017). *Representing Big Data as Networks: New Methods and Insights* (Issue July). http://arxiv.org/abs/1712.09648

Yao, W., Guo, K., Hou, Y., & Li, X. (2022). Hierarchical Structure-Feature Aware Graph Neural Network for Node Classification. *IEEE Access*, *10*, 36846–36855.

https://doi.org/10.1109/ACCESS.2022.3164691

Zhang, H., Li, P., Zhang, R., & Li, X. (2022). Embedding Graph Auto-Encoder for Graph Clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 1–11. https://doi.org/10.1109/tnnls.2022.3158654