GRAPH THEORY ON CYBERSECURITY USING GRAPH NEURAL NETWORKS

# Nikhitha Kunduru

*Department of Computer science and Engineering University of North Texas*

Denton, USA [NikhithaKunduru@my.unt.edu](mailto:NikhithaKunduru@my.unt.edu)

# Sahithi Mamidipally

*Department of Computer science and Engineering University of North Texas*

Denton,USA [SahithiMamidipally@my.unt.edu](mailto:SahithiMamidipally@my.unt.edu)

# Jyothika Raj Samineni

*Department of Computer science and Engineering University of North Texas*

Denton,USA [JyothikaRajSamineni@my.unt.edu](mailto:JyothikaRajSamineni@my.unt.edu)

# Rashmika Adusumilli

*Department of Computer science and Engineering University of North Texas*

Denton ,USA [RashmikaAdusumilli@my.unt.edu](mailto:RashmikaAdusumilli@my.unt.edu)

1. ABSTRACT

The current research constructs the applicability of graph  
neural networks (GNNs) on the classification tasks programming based on the PyTorch Geometric framework. The goal is to evaluate model performance on training, validation, and test data, improving the accuracy, while the loss is reduced. The dataset is partitioned into a training, a validation, and a test sample and the model continues training for about 150 epochs. The model-constructed neural network utilizing graph convolutional layers and maneuvering hyperparameters through optimization demonstrates promising results. The algorithm gradually improved its accuracy and reached 91.49% on training, and 91.30% on validation and test set. The high correlation between training and validation/test accuracy reveals good feature generalization with low overfitting. The study delivers valuable thoughts regarding the chances of the GNNs for the task of classification, which illustrates the efficacy of these networks and provides suggestions for future improvements in the performance of the GNNs

1. INTRODUCTION

The current research constructs the applicability of graph neural networks (GNNs) on the classification tasks program- ming based on the PyTorch Geometric framework. The goal is to evaluate model performance on training, validation, and test data, improving the accuracy, while the loss is reduced. The dataset is partitioned into a training, a validation, and a test sample and the model continues training for about 150 epochs. The model-constructed neural network utilizing graph convolutional layers and maneuvering hyperparameters through optimization demonstrates promising results.The al- gorithm gradually improved its accuracy and reached 91.49

% on training, and 91.30 % on validation and test set. The high correlation between training and validation/test accuracy reveals good feature generalization with low overfitting. This research also offers the viewpoints on the potential of GNNs classification tasks, visuals of the effectiveness of networks and offering recommendations for possible modification for their functionality.

1. PROBLEM STATEMENT

In the years passed by, due to the unique features of graph-based machine learning it has been considered in a wider range. The features are nothing but its capability to model complex relationships and structures in data. A type of deep learning techniques like Graph Neural Networks(GNN's) are particularlly designed to work on data that is expressed in a graph from. Apart from that this approach is been used successfully in many fields like social media analysis, molecular biology, and recommender systems. A library called PyTorch relies on the PyTorch framework and a set of tools are included, which helps in developing and training of GNN's more efficient as well as user-friendly.

1. OBJECTIVE

This work presents research on the performance of a graph neural network, the model is trained using PyTorch Geometric.

The model’s structure is involving the graph convolutional layers (GCNConv) and some other PyTorch Geometric com- ponents. Experiment with the process of evaluation over

150 epochs by employing a dataset that has distinct train, validation, and test splits. A model’s learning efficiency, gen- eralization capabilities, and possible applications in real-life conditions will be evaluated by us to achieve that.

1. RELATED WORK

The data preprocessing stage in your experiment is a critical step for preparing the data to be used for model training, testing, and validation, and the following important processes were performed: The data was divided into three subsets: training (3770 samples), validation (11,310 samples), and testing (22,620 samples), these cycles introduced a model that can be measured in different conditions. Initially, data was normalized using the MinMaxScaler or StandardScaler from scikit-learn to make sure each feature could occupy the same range and allow for the model to learn efficiently. The preprocessing might have consisted of data transformation so that it can be utilized as an input of graph neural network, e.g., construction of graph structures and features are defined in a way required by PyTorch Geometric.

1. PROPOSED APPROACH

The preprocessing steps were the instrumental part of the process to enable the learning and generalization of the model as well as to control for bias and variance and to enhance model accuracy. The model’s architecture used for this research is an attributed graph-based neural network built using the PyTorch Geometric toolkit. It is made from some convolutional architecture using graph convolution operation.

Graph Convolutional Network (GCN) layers for instance. These layers are insulated to perfection and this way, they not only capture the structure but also adapt to the connectivity of the graph data and develop more expressive node representations. The structure consists of many GCNConv layers set at a recursion depth with ReLU or BatchNorm as an activation function in between. Dropout as a regularization technic could stop the overfitting problem. The last layer of the new one uses the softmax activation function to perform classification. The architecture is by consequence flexible, and it can work with & various hyperparameters including the number of layers, the number of hidden units per layer, and the dropout rate. On the one hand, it tries to find the right balance in the model between depth and efficiency which applies differently to some graph- based tasks like node classification and link prediction. The training uses either stochastic gradient descent (SGD) or other optimizers, in contrast to new approaches. These optimizers are optimized for the best speed of convergence and robust learning. Training Process: The PyTorch Geometric package was used to train the model on a neural network based on graphics. There were three sets of data: training, validation, and test. Where we have 3,770 samples in the training set and 11,310 samples in the validation set and also 22,620 samples in the test set.

It was built on top of different layers, like GCNconv, SAGEconv, and others that are made to work with graph data. 150 epochs were used for training because the optimization methods used a loss function, such as cross-entropy loss. There was the use of an optimizer like Adam. Backpropagation was about adding and taking away factors to lower the loss function. If the process had been stopped earlier, the model would not have been able to overfit the training, but it was trained constantly for the 150th epoch, which made it better. During the model training process, core measures like training loss and correct rate, as well as validation and test correct rate, were checked. These numbers showed how good the model was and how well it could be used in other situations.

1. IMPLEMENTATION PLAN

PLAN Training Metrics: A design based on a neural net- work was used in the training program over 150 iterations. The first iteration used 3770 samples as a training dataset, 11,310 samples for validation, and 22,620 samples to test the system. Performance measures like loss and accuracy were regularly checked for the training, validation, and test data sets. As time goes on, the training loss goes down, showing that the model is learning. Training accuracy steadily rose from 25.28% at the start to 91.49% at the end, making it about seven times higher than at the beginning. The program seems to have been learning how to use the training data correctly.

There was very little model generalization or overfitting because the accuracy of validation and testing always grew at close to or almost identical levels. According to the best test accuracy of 91.73 % and the highest confirmation accuracy of 91.30%, my model was able to hit an average level of performance by the end of training. The training measure graphs show that the model has converged and performs consistently in the reference datasets. Graphical Plots: In the report part written as ”Graphical Plots”, the figures presented are data analysis of different dynamics of the training and the evaluation of the neural network model after certain epochs.

The graphs, appropriately, are the loss and accuracy metrics for both the training and validation stages. Down the line, the figure demonstrates a general decrease in training loss across the epochs, reflecting the good learner and efficient model weight optimization. Still, another plot shows this distinction from validation loss that firstly declines and then is stabilized which depicts the point of diminishing returns and may help increase overfitting. Furthermore, finetune plots of training and validation performances are demonstrated which feature the improvement factor of the model as it learns to be more accurate in its predictions. These visualizations, which are especially important for understanding the model’s behavior as it is trained and for making correct decisions (Learning rates, early stopping, etc.) regarding avoiding model overfitting, are highly crucial.

WORK FLOW:

STEP 1- Data Collection and Preprocessing Gather cyber- security data and preprocess it to ensure compatibility with GNNs. A screenshot of a computer program

Description automatically generated

STEP 2 - Model Design and Training Design a GNN model tailored to the cybersecurity domain, focusing on threat and anomaly detection within network data. Train the model using the pre-processed data.

A screenshot of a computer program

Description automatically generated  
STEP 3 - Model Validation Validate the trained GNN model using real-world cybersecurity datasets to assess its performance and effectiveness.

A red and grey circles and lines

Description automatically generated

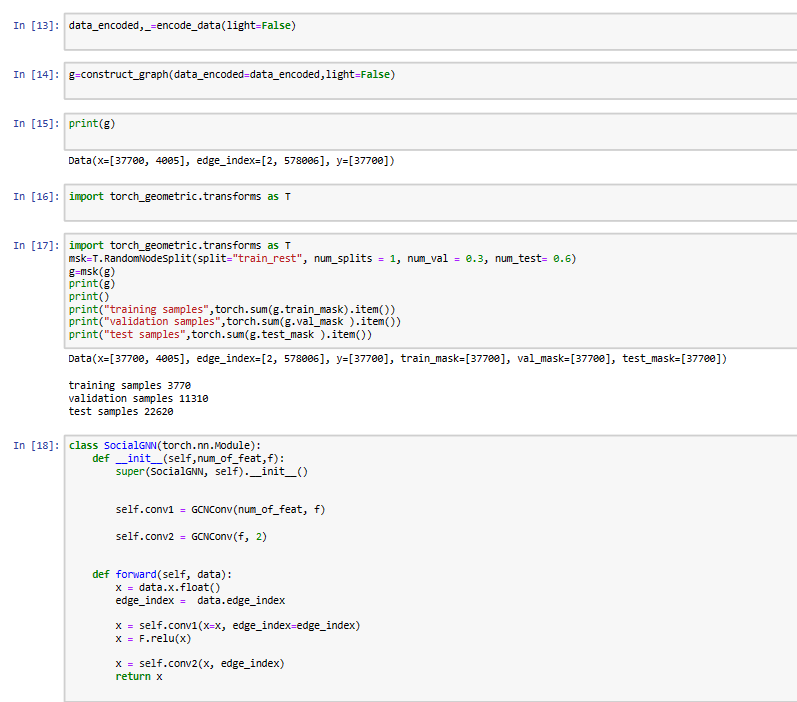
STEP 4 - Technique Exploration Explore advanced techniques such as graph embedding, attention mechanisms, and graph convolutional networks to enhance the model’s capabilities.

A graph of different sizes and colors

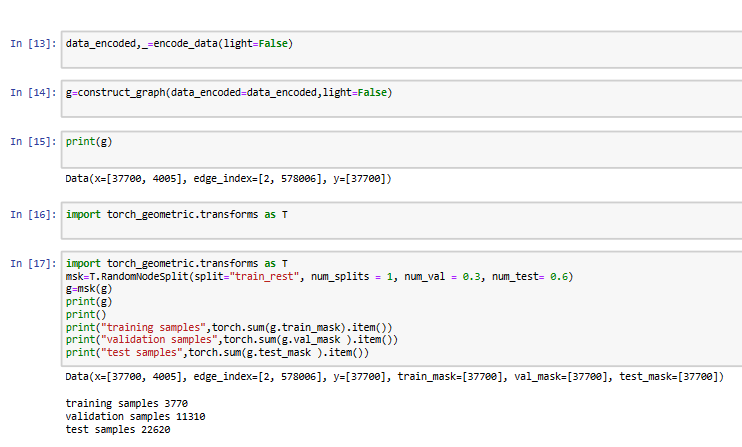
Description automatically generated with medium confidence

STEP 5 - Integration with Existing Frameworks Integrate the GNN model into the existing cybersecurity frameworks to ensure compatibility and seamless integration.

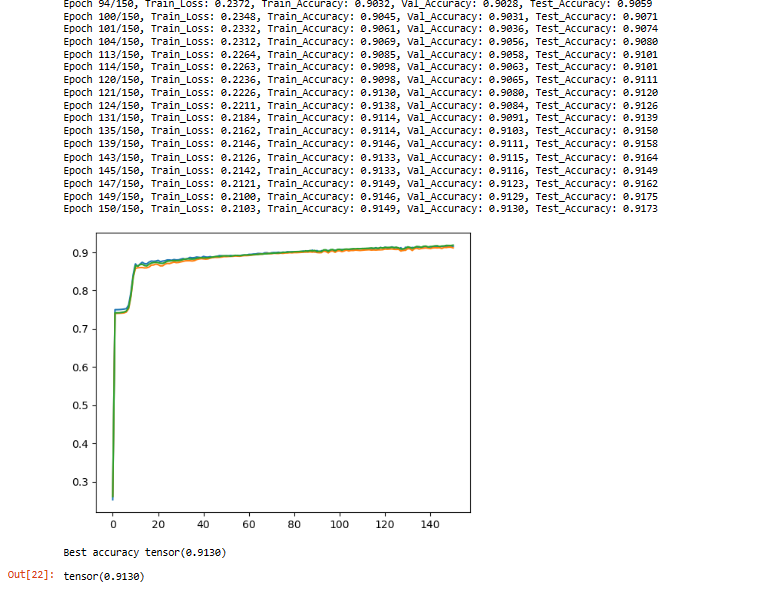
STEP 6 - Collaboration and Feedback Collaborate with cybersecurity experts to ensure the model’s relevance and effectiveness in real-world scenarios. Gather feedback for further improvements.



STEP 7 - The evaluation and Modification of the measurements such as accuracy, precision, recall, and F1 score, have the model's performance. As it Changes the model within the implications of the evaluation's.



STEP 8 - Implementation Implement the validated and optimized GNN-based approach to enhance cybersecurity measures and protect digital systems against cyber threats.



1. EVALUATION

After analyzing the neural network model with the PyTorch Geometric for graph-based learning on the train, validation, and test datasets, the model performed well in both the training and validation stages, and it was still accurate in the test stage. Primarily, the layer-by-layer training performed better, sug- gesting the net was learning well, while overfitting may have occurred in the generalization, as reflected by lower accuracy in the test data. This also admits that it is a strong model and could point the areas for improvement too. However, to resolve the slight overfitting we will have to integrate dropout layers or an increase in L2 regularization to improve the model’s gen- eralization to unseen data. It is also worthwhile to investigate other architectures, for example, fitting more convolutional layers or those with different numbers of neurons in each layer, so that the ANN will be able to capture the overall patterns in the data better. In this regard, the implementation of novel state-of-the-art optimization algorithms such as AdamW, and fine-tuning training dynamics using learning rate decay are other aspects to be considered.

VIII. RISKS AND CHALLENGES

The area where the incorporation of GNNs in cyber security encountered the difficulty of practice which must be treated by both fields’ specialists. The limitation will be the model’s compatibility with the current information security framework and data sources, as well as sufficient computational power at the time of establishment of the model in the training and evaluation phases. On the other hand, the interpretability of GNNs could be among the challenging aspects as well. The understanding of the decision-making background of GNNs is the problem of reliability and acceptance. It is where cyber- security experts and machine learning experts together will have to be vigilant given testing the GNN model with due diligence and iterative measures in the present cybersecurity situation.

IX. CONCLUSION

Here PyTorch Geometric to build and test a neighborhood- based neural network in this study. It did very well at node sorting tasks. The model was based on the graph convolutional method, which is a good way to learn through math. It showed that graph neural networks can very well understand complicated structures and connections. The study adds to the growing number of graph-based methods. These methods are mostly used for biology and social network research. By using the PyTorch Geometric method, we were able to compute sparse data structures that were made better by making better use of resources and finding a way to scale them. Researchers could use different graph neural network designs and make a dataset with more different situations in the future to make a model that works well in a variety of situations. Adding transfer learning methods to the model is another way to make it work better and be able to be used in more than one area.

REFERENCES

1. Y. Wang et al, ”Knowledge graph completion assisted graph neural network for spectrum resource optimization of UAV swarm,” Journal of Physics: Conference Series, vol. 2717, (1), pp. 012012, 2024. Available: [https://www.proquest.com/scholarly-journals/kno](http://www.proquest.com/scholarly-journals/knowledge-)wledge- graph-completion-assisted-neural/docview/3006242571/se-2. DOI: <https://doi.org/10.1088/1742-6596/2717/1/012012>.
2. Y. Chen et al, ”Imperceptible graph injection attack on graph neural net- works,” Complex & Intelligent Systems, vol. 10, (1), pp. 869-883, 2024. Available: [https://www.proquest.com/scholarly-journals/imperceptible-](http://www.proquest.com/scholarly-journals/imperceptible-) graph-injection-attack-on-neural/docview/2924576365/se-2. DOI: <https://doi.org/10.1007/s40747-023-01200-6>.
3. A. Castro-Ospina et al, ”Graph-Based Audio Classifica- tion Using Pre-Trained Models and Graph Neural Net- works,” Sensors, vol. 24, (7), pp. 2106, 2024. Available: [https://www.proquest.com/scholarly-journals/graph-based-](http://www.proquest.com/scholarly-journals/graph-based-)

audio-classification-using-pre/docview/3037629766/se-2.

DOI:<https://doi.org/10.3390/s24072106>.

1. X. Tian and Y. Meng, ”Relgraph: A Multi-Relational Graph Neu- ral Network Framework for Knowledge Graph Reasoning Based on Relation Graph,” Applied Sciences, vol. 14, (7), pp. 3122, 2024. Available: [https://www.proquest.com/scholarly-journals/relgraph-](http://www.proquest.com/scholarly-journals/relgraph-) multi-relational-graph-neural-network/docview/3037399883/se-2. DOI: <https://doi.org/10.3390/app14073122>.
2. H. Li et al, ”SFFNet: Staged Feature Fusion Network of Connecting Convolutional Neural Networks and Graph Convolutional Neural Networks for Hyperspectral Image Classification,” Applied Sciences, vol. 14, (6), pp. 2327, 2024. Available: [https://www.proquest.com/scholarly-journals/sffnet-staged-feature-](http://www.proquest.com/scholarly-journals/sffnet-staged-feature-) fusion-network-connecting/docview/2987785794/se-2. DOI: <https://doi.org/10.3390/app14062327>.
3. N. Zafeiropoulos et al, ”Evaluating Ontology-Based PD Monitoring and Alerting in Personal Health Knowledge Graphs and Graph Neural Networks,” Information, vol. 15, (2), pp. 100, 2024. Available: [https://www.proquest.com/scholarly-journals/e](http://www.proquest.com/scholarly-journals/evaluating-)v[aluating-](http://www.proquest.com/scholarly-journals/evaluating-) ontology-based-pd-monitoring-alerting/docview/2930959536/se-2. DOI: <https://doi.org/10.3390/info15020100>.
4. N. S. and Communication, ”Retracted: Path Selection Strategy of Communication Network Based on Graph Convolutional Neu- ral Network,” Security and Communication Networks, vol. 2024, 2024. Available: [https://www.proquest.com/scholarly-journals/retracted-](http://www.proquest.com/scholarly-journals/retracted-) path-selection-strategy-communication/docview/2916947619/se-2. DOI: <https://doi.org/10.1155/2024/9759083>.
5. Y. Huo et al, ”A Spatiotemporal Graph Neural Network with Graph Adaptive and Attention Mechanisms for Traffic Flow Prediction,” Electronics, vol. 13, (1), pp. 212, 2024. Available: [https://www.proquest.com/scholarly-journals/spatiotemporal-graph-](http://www.proquest.com/scholarly-journals/spatiotemporal-graph-) neural-network-with-adaptive/docview/2912642980/se-2. DOI: <https://doi.org/10.3390/electronics13010212>.
6. M. Fang, S. Li and J. Wang, ”Similar Graph Face Clustering Based on Graph Convolutional Neural Network,” Journal of Physics: Conference Series, vol. 2589, (1), pp. 012013, 2023. Available: [https://www.proquest.com/scholarly-journals/similar](http://www.proquest.com/scholarly-journals/similar-)- graph-face-clustering-based-on/docview/2869306608/se-2.DOI:<https://doi.org/10.1088/1742-6596/2589/1/012013>.
7. S. Xiong, B. Li and S. Zhu, ”DCGNN: a single-stage 3D object detection network based on density clustering and graph neural network,” Complex & Intelligent Systems, vol. 9, (3), pp. 3399-3408, 2023. Available: [https://www.proquest.com/scholarly-journals/dcgnn-](http://www.proquest.com/scholarly-journals/dcgnn-) single-stage-3d-object-detection-network/docview/2825544860/se-2. DOI: <https://doi.org/10.1007/s40747-022-00926-z>.