Graph Neural Network Classification Architectures for Mutagenic Molecule Detection

Manos Chatzakis
emmanouil.chatzakis@epfl.ch

ABSTRACT

This report presents an evaluation of different Graph Neural Network (GNN) architectures for the task of mutagenic molecule detection through binary classification. We experiment and assess different convolution architectures suitable for graph-represented molecule data, and develop methods to incorporate both node and edge features of a network. Our experiments demonstrate the superiority of GraphSAGE architecture, which manages to achieve up to 79% accuracy on balanced test data of the MUTAG dataset, using the node features of the graph.

1 INTRODUCTION

Motivation. Nowadays, Graph Neural Networks (GNNs) are pivotal in graph classification research across every scientific and social domain[10][4][5]. Over the recent years, their representation power has found applications in biomedicine, especially in toxicology and drug discovery research. A specific use case is mutagenic molecule detection, where molecules are represented as a graph, with nodes representing atoms, and edges the chemical bonds between them. Those graphs can be provided to GNN models to classify the molecule as mutagenic or not.
Challenge. The main challenge in mutagenic molecule detection is that the best-performing Graph Neural Network architecture is tightly connected to the graph structure, making it difficult to distinguish which architecture to use in every case. In addition, although the graph data include node and edge features for each atom and chemical bond, most GNNs can't incorporate the edge features in the computations.
Approach. In this report we aim to tackle the aforementioned challenges and provide a robust classification GNN model. We begin by extensively evaluating different architectures, tuning all hyperparameters to find which model performs best using the node features. Then, we describe a solution that includes the edge features by concatenating to the node features of each node an aggregated representation of the edge features that each node participates in. Then, we re-evaluate our best-performing model using this enhanced feature data. Our findings demonstrate that for the molecules of the MUTAG dataset, the best accuracy of 79% is obtained by a 2-layer GraphSAGE model using node features.

2 GRAPH CONVOLUTION LAYERS

In this section, we describe the convolution layer architectures we implemented.
Normal Convolution. Normal Graph Convolution layers perform a forward pass by averaging the features of the neighborhood of each node [9].
Attention. Graph Attention forward pass is defined as ${h_v}^{l+1} = \sigma\left(\sum_{u\in N(v)\cup\{v\}} {a^l}_{vu}{h'_u}^{l}\right)$ with ${h'_v}^{l} = W_v {h_v}^{l}$, where a^l_{vu} is the normalized attention weight between node v and its neighbors u and itself, computed as described in [7]
GraphSAGE. GraphSAGE layers [2] aggregate the features of the neighborhood of each node, and then concatenate the result with the node features. As an aggregation module we use Max Pooling [6].

3 DATASET

Here, we describe our dataset structure and data preprocessing.
Description. We use the MUTAG[3] dataset, a biomedical collection of 188 graph-represented molecules. Each graph is a collection of nodes (atoms) and edges between them (chemical bonds), and there is an associated label with each graph indicating if the molecule is mutagenic or not. We use 70% of the data for training, 15% for validation, and 15% for testing. Each MUTAG graph node contains 7 features, while each edge has 4 features.
Data Balancing. Out of the 188 graphs, 125 of them are labeled as mutagenic and 63 as non-mutagenic. This imbalance can lead to performance degradation, as the model will encounter imbalanced sets of examples during training. Thus, we randomly discarded some mutagenic graphs, normalizing this severe imbalance. We evaluated our models with the unbalanced set too, but this decision led to unstable results, even when we used weighted performance metrics.

4 CLASSIFICATION USING NODE FEATURES

In this section, we describe our process to select the most appropriate architecture for classification using node features, for different configurations of the layers of section 2.
Layers and Learning Rate. The first step to deciding what the best architecture is, is to determine how many layers to use and the output size of each layer, for every layer type. We begin by heuristically choosing a set of different layer numbers and output features for each architecture, to determine the range of meaningful values to use. We concluded that the best results for each architecture were for 1-3 layers with output features in the range of 4-256. Then, we configured several models per architecture for systematic evaluation. Regarding the Learning Rate (*LR*), we used scheduling to gradually decrease the value of *LR* as the epochs progressed, to achieve faster convergence, by doing large steps initially, and smaller ones during the last epochs. However, the most robust and best-performing method was to use a constant small *LR* value, because it allowed the models to progress to an optimal solution steadily. After setting the hyperparameters, we trained for 300 epochs and evaluated the accuracy of the test data. The results are depicted in figure 1. The naming convention used is Model-LayerOutputSize. We concluded that the best-performing model for normal convolution is Conv-256-16-8 (accuracy 63%), Attn-16-4 (accuracy 68%), and GraphSAGE the Sage-128-8 (accuracy 79%) for each type of layer architecture.
Evaluation. To define the final model, we used the best versions of each architecture described above, along with combinations of them (The naming convention indicates the order of the layers used). The overall evaluation of the models in the test data is presented in figure 2. The best-performing model is GraphSAGE-128-8, achieving 79% accuracy and 75% f1-score.
Discussion. Our experiments concluded that 2-layer GraphSAGE outperforms the alternative architectures. This is because of the generalization power of GraphSAGE to the heterogeneous graphs of MUTAG and the aggregation mechanism of the model. This heterogeneous nature and structure of MUTAG graphs also explain

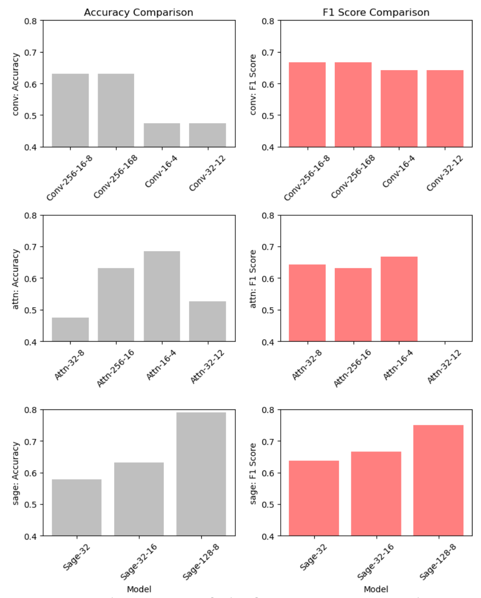

Figure 1: Evaluation of different GNN architectures

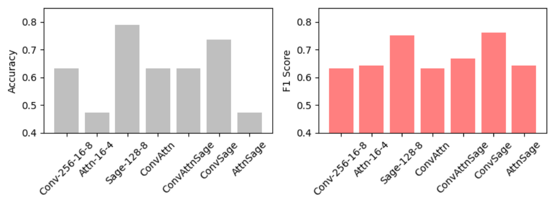

Figure 2: Final GNN model comparison

the lower performance of the Normal Convolution and Attention. It is important to note that we tried the experiments with the unbalanced data. The f1-score was 80% for some models, but the accuracy was significantly lower. In addition, the severe imbalance of the training examples could hurt the generalization power of the models and thus we decided to keep the balanced data results.

5 CLASSIFICATION INCORPORATING EDGE FEATURES

The approach of section 4 is agnostic to the edge features of the dataset. Here, we describe a method that incorporates the available edge features of MUTAG, and we discuss its performance.

Approach. The core idea of our approach is to be able to enhance the node features using the edge features for each node. The main challenge is that edge features naturally correspond to the edges of the graph, rather than the nodes, thus we can not directly incorporate them in any meaningful way into the node features using concatenation or aggregation. Our solution springs from the realization that the features of an edge belonging to a node may contain useful information about the node itself. As edges in our scenario represent chemical bonds, the information related to those compounds might be able to describe the nodes belonging to this connection.

Our method utilizes the adjacency matrix to locate the edges of each node and collects the edge features of the edges that each node participates in. This leads to a generation of an $NE \times E$ matrix for each node, where NE is the number of the edges of the node, and E is the number of edge features in MUTAG. This matrix contains useful information about each node, based on which chemical links it corresponds, and thus it would make sense to incorporate this matrix into the node features. However, combining this per-node matrix is not a straightforward task, as we need to transform it into dimensions feasible for combinations with the node features. For this reason, we use an $AGGREGATION$ function (mean aggregation) to transform the $NE \times E$ to a $1 \times E$ matrix for each node. Now we can concatenate this matrix to the node features and create an enhanced, $N \times (E+F)$ matrix, where N is the number of nodes and F is the number of node features in MUTAG. Using this approach, we managed to generate a new node feature matrix, which is enhanced by the information contained in the edges of each node.

Evaluation. We used the 2-layer GraphSAGE model from section 4. We trained using the same pipeline described before, but now using the enhanced feature matrix. We used the same model architecture, as the only technical difference was the number of features, which had no impact on how the layers operate. The training summary is depicted in figure 3. We see that the model managed to converge after 300 epochs, but unfortunately, we achieved an accuracy of 63% and an f1-score of 66% on the testing data, not improving the performance. In addition, re-configuring the approach with different $AGGREGATION$ functions, hyperparameters, and architectures, led to worse performance.

Discussion. The worse performance is the aftereffect of the assumption that the edges of each node provide useful insights about the node itself. This is a dataset-specific assumption that seems to not hold for MUTAG, as our approach does not improve the quality of the features. In addition, an important factor of the evaluation is the absence of more data samples. Training and testing on more data could provide more insights about the usefulness of our method. Overall, the research question of incorporating edge features in the computations is an evolving field of GNNs[8][1], with state-of-the-art methods trying to extend the convolution architectures to efficiently take into account both node and edge features.

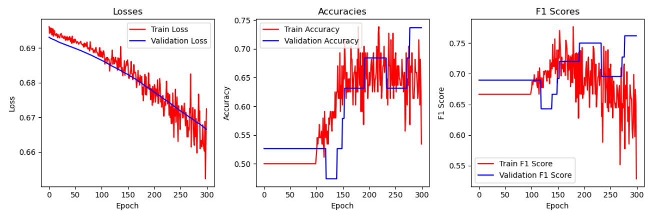

Figure 3: Training summary of GraphSAGE using Edge Features

6 CONCLUSION

We presented a biomedical graph classifier for mutagenic molecule detection that uses a 2-layer GraphSAGE GNN, able to incorporate both node and edge features in the computations. Our experimental evaluation demonstrates that the classifier using a 2-layer GraphSAGE network with node features achieves 79% accuracy and 75% f1-score of the data of the MUTAG dataset.

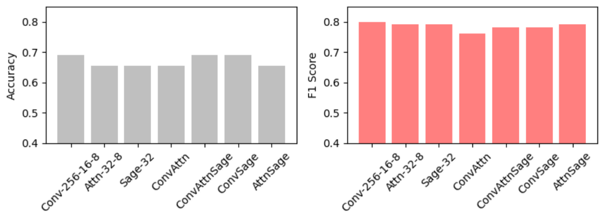

Figure 4: (Appendix) Final GNN model comparison (unbalanced data)

REFERENCES

[1] Liyu Gong and Qiang Cheng. 2019. Exploiting edge features for graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 9211–9219.

[2] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[3] Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. 2016. Benchmark Data Sets for Graph Kernels. http://graphkernels.cs.tu-dortmund.de

[4] Michelle M Li, Kexin Huang, and Marinka Zitnik. 2022. Graph representation learning in biomedicine and healthcare. *Nature Biomedical Engineering* 6, 12 (2022), 1353–1369.

[5] Rui Li, Xin Yuan, Mohsen Radfar, Peter Marendy, Wei Ni, Terence J O'Brien, and Pablo M Casillas-Espinosa. 2021. Graph signal processing, graph neural network and graph learning on biological data: a systematic review. *IEEE Reviews in Biomedical Engineering* (2021).

[6] Naila Murray and Florent Perronnin. 2014. Generalized max pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2473–2480.

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

[8] Yulei Yang and Dongsheng Li. 2020. NENN: Incorporate Node and Edge Features in Graph Neural Networks. In *Proceedings of The 12th Asian Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 129)*, Sinno Jialin Pan and Masashi Sugiyama (Eds.). PMLR, 593–608. https://proceedings.mlr.press/v129/yang20a.html

[9] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. 2019. Graph convolutional networks: a comprehensive review. *Computational Social Networks* 6, 1 (2019), 1–23.

[10] Xiao-Meng Zhang, Li Liang, Lin Liu, and Ming-Jing Tang. 2021. Graph neural networks and their current applications in bioinformatics. *Frontiers in genetics* 12 (2021), 690049.

7 APPENDIX

We provide the results of the models with the unbalanced data, presented in figure 4.