Deep Learning Approaches for Mutagenicity Prediction in Chemical Compounds

Jonas-Mika Senghaas, *mika.senghaas@epfl.ch*
Deep Learning in Biomedicine (CS-502), EPFL, Switzerland

***Abstract*—This project explores the use of graph convolutional networks (GCN) to predict mutagenicity in chemical compounds. Through a rigorous evaluation of various graph convolutional and pooling techniques on the MUTAG dataset, we have found that GCNs can effectively learn graph representations, achieving high accuracy in mutagenicity prediction. The GraphSAGE convolutional layer with sum aggregation outperforms attention-based and edge graph convolution, with different pooling operations showing minimal changes in the downstream performance.**

I. Introduction

Mutagenic compounds have the potential to induce genetic mutations in living organisms, elevating the risk of cancer. Detecting mutagenicity is thus a critical endeavour in the field of biomedicine. This project investigates the capacity of Graph Convolutional Networks (GCNs) with diverse convolutional and pooling configurations for this purpose. All experiments conducted are fully reproducible and openly available on GitHub.

II. Data

The MUTAG dataset [1] is a popular benchmark dataset for graph classification in the biomedical domain. It consists of 188 graphs, each representing a chemical compound with nodes being atoms and edges being bonds. Both node and edge types are labelled as one of seven atom types and four bond types, respectively. Each chemical compound is labelled mutagenic (positive class) or non-mutagenic (negative class). An example of the data is shown in Figure 3 in the Appendix. Table I displays statistics of the entire dataset. The dataset consists of more positive (66%) than negative examples (34%). Further, all graphs are fully-connected and small, with an average of ≈ 18 nodes, ≈ 20 edges and an average degree of 2.19. The mutagenic graphs tend to be slightly larger with a higher average node and edge count.

	Overall	**Positive** (1)	**Negative** (0)
#Graphs	188	125 (66%)	63 (34%)
Avg. #Nodes	17.93	19.94	13.94
Avg. #Edges	19.79	22.40	14.62
Avg. Degree	2.19	2.24	2.09
Avg. Connectivity	1.0	1.0	1.0

Table I
Dataset Statistics

III. Methodology

The objective is to accurately predict the mutagenicity of chemical compounds by learning both from the graph's topology and its node and edge features. This study investigates the potential of graph convolutional networks for this task. Architecturally, all models are feed-forward graph neural networks that iteratively update a graph's node feature representation into a meaningful latent space. To perform the final graph prediction a global pooling operation maps the node representations to a single graph representation. Finally, a linear layer with a sigmoid activation function outputs the probability of a graph being mutagenic. This project explored four different graph convolutional layers and two global pooling operations, which are introduced in the following.

A. Graph Convolution Layers

Three of the four graph convolutional layers are well-known methods, including GraphConv [2], GraphSAGE [3] (with sum aggregation), and GAT [4]. The specifics of the methods are not discussed here. For details, please refer to the respective papers. The fourth layer is an adapted version of the edge convolutional EGNN(C) layer, introduced by Gong et al. [5]. Unlike the other approaches, it incorporates edge features to update the node representations. The core idea is to treat the edge feature matrix $\mathbf{E}$ as a collection of P adjacency matrices $\mathbf{E}_{..p} \forall p \in P$. Gong et. al propose to normalise (Eq. 2 & 3) the edge feature matrix and then perform a graph convolution on each of the P channels. The results are then multiplied element-wise and fed through a non-linear activation function. However, as the edge features in this dataset are one-hot encoded, the element-wise product will lead to sparse matrices, which limit the learning abilities of the model. To counteract this, element-wise summation is used. Furthermore, each channel is given its own learnable weight matrix $\mathbf{W}_p$ to allow the model to learn different representations for each edge type. The layer can be expressed as

$$\mathbf{H}^{(l+1)} = \sigma\left(\sum_{p=1}^{P} \tilde{\mathbf{E}}_{..p}\mathbf{H}^{(l)}\mathbf{W}_p^{(l)}\right), \tag{1}$$

where $\mathbf{W}_p^{(l)}$ is a trainable weight matrix in the l-th layer for the p-th channel of the edge feature matrix, σ is a non-linear activation function and $\mathbf{H}^{(l)}$ is the node feature matrix of the l-th layer.

B. Global Pooling Layers

To obtain a fixed-sized representation of the graph, a global pooling operation is applied to the latent node representations after the graph convolutional layers. This study considers the standard mean and max pooling operations.

IV. Experiments

The study aims to explore the potential of graph convolutional networks for mutagenicity prediction, as well as the impact of different graph convolutional and pooling layers on the model's performance. To this end, a grid search was performed over a set of model and training hyper-parameters. For each combination of the four graph convolutional layer types (GraphConv, GraphSAGE, GAT, EdgeConv) and two global pooling layers (Mean, Max), models of varying sizes were trained and evaluated with varying training hyper-parameters. Model sizes changed by altering the number of layers ($L \in \{3, 5, 7\}$) and the hidden units per layer ($H \in \{8, 32, 64\}$).

All 72 models were trained using stochastic gradient descent with a batch size of 1 for learning rates of $\gamma = \{0.01, 0.001\}$ and a fixed number of $\eta = 100$ epochs. Each model was optimised under weighted, binary cross-entropy objective using the Adam optimiser [6]. A split of 75% is used for training, 15% for validation and 10% for testing. To obtain a balanced classifier, model selection is performed based on the validation Macro F1 score which averages the F1 score over both classes.

V. Results

Table II displays the validation performance of the five best models. All models achieve a high validation accuracy and Macro F1 score of $\sim 90\%$. All models use the GraphSAGE convolutional layer with sum aggregation. However, the number of layers, hidden units and pooling operation differ.

Parameters					Validation	
L	D	γ	Conv	Pool	Acc. (%)	M.-F1 (%)
5	64	1e-2	GraphSAGE	Max	92.71	92.86
7	32	1e-2	GraphSAGE	Max	89.16	89.29
3	32	1e-1	GraphSAGE	Max	89.16	89.29
3	32	1e-1	GraphSAGE	Mean	88.93	89.29
5	64	1e-1	GraphSAGE	Mean	88.93	89.29

Table II
Validation Performance of Top 5 Models

Figure 1 shows the validation Macro F1 score of all models as a function of the number of trainable model parameters. The convolutional and pooling layer types are encoded through colour and shape, respectively. In line with the inspection of the top 5 models, the figure shows that the best performing models are all GraphSAGE models. Interestingly, neither the attention-based nor the edge convolutional layer are able to reach the same performance. The training of attention-based models was found to plateau after a certain number of epochs, which is reflected in the lower validation performance. The edge convolutional models often overfit to the training data and not generalise well to the validation data as a result. Further, the scatter plot does not suggest any clear correlation between the number of trainable parameters or the pooling operation and the final performance.

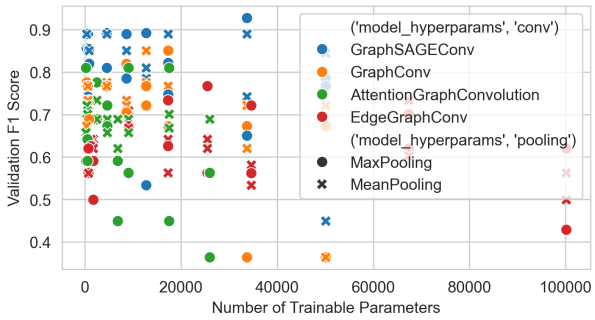

Figure 1. Validation Macro F1 Score as a function of Model Size

Figure 2 breaks down "model size" into the number of layers and hidden units through a heat map. The figure highlights that more complex models do not significantly improve performance. The best performing models are relatively simple with $L = 3$ layers and $H = \{32, 64\}$ hidden units.

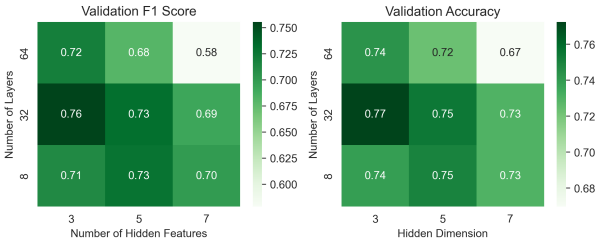

Figure 2. Average Macro F1 Score for Different Numbers of L and H

Finally, evaluating the best-performing model on the test set yields an accuracy of $\sim \mathbf{89}\%$ and a Macro F1 score of $\sim \mathbf{87}\%$, proving the model's ability to generalise.

VI. Summary

The experiments show that graph convolutional networks learn powerful representations of chemical compounds that allow for accurate prediction of mutagenicity. Given the small data size, smaller architectures were found to generalise better and are therefore preferred. The results suggest that the GraphSAGE layer creates the best models, while the choice of the global pooling layer does not seem to have a significant impact on the performance.

References

[1] K. Kersting, N. M. Kriege, C. Morris, P. Mutzel, and M. Neumann, "Benchmark data sets for graph kernels," 2016. [Online]. Available: http://graphkernels.cs.tu-dortmund.de

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017.

[3] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2018.

[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2018.

[5] L. Gong and Q. Cheng, "Exploiting edge features in graph neural networks," 2019.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

VII. Appendix

A. Graph Visualisation

A visualisation of the graph structure of a mutagenic and non-mutagenic graph. In addition to the graph topology, the plot shows the integer-encoded node and edge types and denotes the graph's mutagenicity with the node colour.

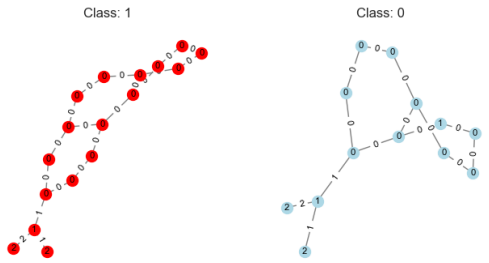

Figure 3. Mutagenic vs. Non-Mutagenic Graph

B. Training Curves

Figure 4 displays the training curve of the best model when trained on the full training set. The figure shows the training and validation loss (left), accuracy (middle) and Macro F1 score (right) over the $\eta = 100$ epochs. The model converges after approximately 50 epochs.

C. Doubly Stochastic Normalisation

Gong et al. [5] propose a doubly stochastic normalisation of the edge feature matrix to prevent the node feature representation from exploding. The normalisation happens in two steps: The matrix is first normalised row-wise (Eq. 2) and then column-wise (Eq. 3). The resulting matrix is doubly stochastic, i.e. the sum of each row and column is equal to 1 with all non-negative entries.

$$\tilde{\mathbf{E}}_{i,j,p} = \frac{\mathbf{E}_{i,j,p}}{\sum_{k=1}^{N} \mathbf{E}_{ikp}} \tag{2}$$

$$\tilde{\mathbf{E}}_{i,j,p} = \sum_{k=1}^{N} \frac{\tilde{\mathbf{E}}_{i,k,p} \tilde{\mathbf{E}}_{j,k,p}}{\sum_{v=1}^{N} \mathbf{E}_{vkp}} \tag{3}$$

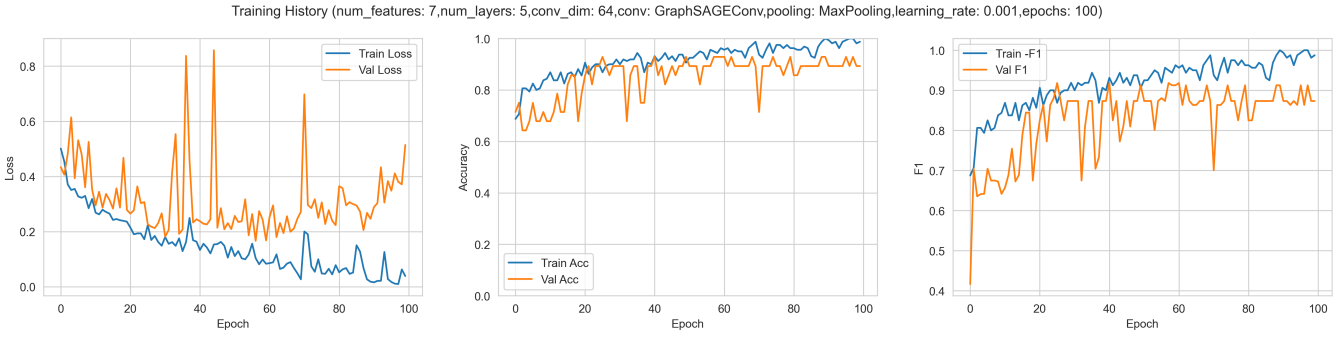

Figure 4. Example Training Curve