## Question 3

**(a)** For this project, we compare two model architectures in terms of their ability to capture correlation between the input features and the target labels in an inductive node classification task. The first studied architecture is GATv2 ($\mathcal{A}$) [Brody et al., 2022], consisting of GATv2 layers interleaved with nonlinearities. The second studied architecture ($\mathcal{B}$) is an extension thereof, consisting of a slightly modified type of layers interleaved with nonlinearities:

$$\mathcal{A}_{\text{layer}} : \begin{cases} e_{ij}^{(t)} = \boldsymbol{a}^{(t)^T} \text{LeakyReLU}\left(\begin{pmatrix} W_l^{(t)} & W_r^{(t)} \end{pmatrix} \begin{pmatrix} \boldsymbol{h}_i^{(t-1)} \\ \boldsymbol{h}_j^{(t-1)} \end{pmatrix}\right) & \text{for } j \in \mathcal{N}_i \\ \alpha_{ij}^{(t)} = \text{softmax}_j(e_{ij}^{(t)}) \\ \boldsymbol{h}_i^{(t)} = \sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(t)} W_r^{(t)} \boldsymbol{h}_j^{(t-1)} \end{cases} \tag{85}$$

$$\mathcal{B}_{\text{layer}} : \begin{cases} e_{ij}^{(t)} = \boldsymbol{a}^{(t)^T} \text{LeakyReLU}\left(\begin{pmatrix} W_l^{(t)} & W_r^{(t)} \end{pmatrix} \begin{pmatrix} \boldsymbol{h}_i^{(t-1)} \\ \boldsymbol{h}_j^{(t-1)} \end{pmatrix}\right) & \text{for } j \in \mathcal{N}_i \\ \alpha_{ij}^{(t)} = \text{softmax}_j(e_{ij}^{(t)}) \\ \boldsymbol{h}_i^{(t)} = W_l^{(t)} \boldsymbol{h}_i^{(t-1)} + \sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(t)} W_r^{(t)} \boldsymbol{h}_j^{(t-1)} \end{cases} \tag{86}$$

My motivation for this study is of a practical nature. In the GATv2 paper, I observed that (in their default implementation) $\boldsymbol{h}_i^{(t-1)}$ influences $\boldsymbol{h}_i^{(t)}$ solely through $e_{ij}^{(t)}$. Considering that many graphs exhibit a correlation between the node targets and the input features, it seemed to me that the model would benefit from a more explicit connection. Therefore, I devised $\mathcal{B}$ as a natural extension of $\mathcal{A}$ that does not introduce any new parameters and I hypothesized that it would be better able to capture the aforementioned input-target correlation.

**(b)** As both architectures are quite convoluted, it is hard to prove any theoretical bounds on their expressivity, certainly considering the three page limit of this project. Therefore, we opt for an empirical setup that consists of a dataset of synthetically generated graphs, where each graph has a known correlation between the input features and the targets. One-layer and two-layer models are trained on this dataset and evaluated on test graphs unseen during training, as the task is inductive. We use this setup to test the hypothesis that $\mathcal{B}$ is better able than $\mathcal{A}$ at classifying nodes correctly when there is a large input-target correlation. All code from this study is available at the following anonymized Git repository: https://anonymous.4open.science/r/GRL-mini-project-2023-7C24

**(c)** The node classification dataset consists of 10 sections, where each section contains 150 training graphs, 50 validation graphs and 50 test graphs. All graphs are undirected, have between 20 and 30 nodes, have no self-loops and were synthetically generated according to the Erdős-Rényi model with $r = 0.2$ [Erdős et al., 1960]. Graphs containing isolated nodes were excluded through rejection sampling, as $\mathcal{A}$ can generate no representations for such nodes apart from the zero vector. The per-node targets are uniformly chosen among 5 color classes, and the initial features are uniformly chosen vectors from

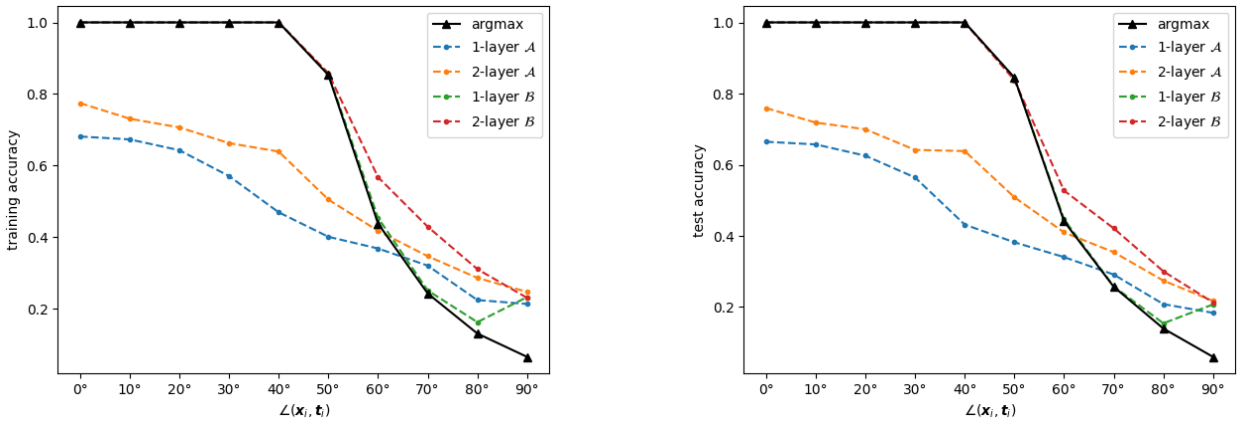
$$\left\{ \boldsymbol{x}_i \in \mathbb{R}^5 \mid \arccos \frac{\boldsymbol{x}_i^T \boldsymbol{t}_i}{\|\boldsymbol{x_i}\| \, \|\boldsymbol{t_i}\|} = \theta_s, \|\boldsymbol{x}\| = 1 \right\} \tag{87}$$

where $\boldsymbol{t}_i$ is the target for node $i$ and $\theta_s$ is the angle specific to the database section $s$. The angles for the different sections are $\{10^\circ \cdot s \mid 0 \leq s < 10\}$ and their cosines correspond with the correlation between $\boldsymbol{x}_i$ and $\boldsymbol{t}_i$ [Rodgers and Nicewander, 1988].

In this study, we compare four models: a one-layer and a two-layer model of each architecture. Call these models $A_1, A_2, B_1$ and $B_2$. For each section of the dataset, all four models are trained on the training graphs and evaluated on the test graphs. The validation graphs are used to monitor the training process and apply early stopping when the validation loss has not improved for 10 epochs. This process is repeated three times. The used hyperparameters are summarized in Table 2, most of these are adopted from the annotated implementation in [Brody et al., 2022] that applies a GATv2 model on the Cora dataset. I believe these hyperparameters are relevant, as Cora is another node-level classification task. The main deviations in this study are the removal of the dropout layer, which is inappropriate because many nodes have low degree, and the smaller size of the models.

Table 2: Hyperparameters

| **Hyperparameter** | **Value** |
|---|---|
| Loss Function | Cross-Entropy |
| Optimizer | Adam |
| Learning Rate | $5 \times 10^{-3}$ |
| Weight Decay | $5 \times 10^{-4}$ |
| Dropout | 0 |
| Max Epochs | 1000 |
| Patience | 10 |
| Batch Size | 15 |
| Hidden Layer Dimension | 5 |
| Inter-layer Nonlinearity | Tanh |




(a) Training Accuracies

(b) Test Accuracies

Figure 1: Training and Test Accuracies

**(d)** Figure 1 plots the median training and test accuracies after training three instances of each model for every section of the dataset, and compares them to a baseline model *argmax*. This baseline predicts that a node belongs to the color class with the same index as the largest entry in its initial feature vector, and can classify nodes with 100% accuracy as long as $\theta = \angle(\boldsymbol{x}_i, \boldsymbol{t}_i) < 45°$.

The test accuracies confirm the hypothesis that $\mathcal{B}$ is better than $\mathcal{A}$ at capturing input-target correlation. Both $B_1$ and $B_2$ match the accuracy of *argmax* for $\theta \leq 50°$. $B_2$ even improves upon it for higher $\theta$, possibly by filtering for the feature vector entry that equals $\cos(\theta)$. In contrast, both $\mathcal{A}$ models are clearly incapable of learning the identity function: even when the targets are identical to the features ($\theta = 0$), they struggle to classify the nodes correctly. The fact that the training accuracies closely match the test accuracies shows that this is indeed an expressiveness issue rather than a generalisation issue. While increasing the hidden layer dimension may enhance the performance of the 2-layer $\mathcal{A}$ model, the B models exhibit a notable superiority even without additional parameters.

These results can be theoretically motivated by considering the one-layer parametrization of $\mathcal{B}$ with parameters $\boldsymbol{a}^{(1)} = \mathbf{0}^d, W_l^{(1)} = I_d$ and $W_r^{(1)} = \mathbf{0}_{d\times d}$. This parametrization implements the identity operation on the node features, which after taking the maximum entry as its predictions amounts to the *argmax* model. Hence, it is not surprising that $B_1$ performs at least as well as the baseline. Further, $B_2$ is provably strictly more expressive than $B_1$: it can use any parametrization of $B_1$ as a first layer, apply a monotone nonlinearity and then apply the aforementioned identity layer to obtain the same classifications as the used $B_1$ parametrization. Therefore, it is in line with expectations that $B_2$ further outperforms $B_1$.

For the $\mathcal{A}$ models, there is not such a straightforward parametrization of the identity operation. Instead, $A_1$ must learn to manipulate the $e_{ij}$ such that $\sum_{j\in\mathcal{N}_i} \alpha_{ij}^{(t)} W_r^{(t)} \boldsymbol{x}_j$ has the same maximal element as $\boldsymbol{x}_i$, for all $i$. This is a much harder task, both intuitively and evidenced by the results. Thanks to its second layer, $A_2$ has the additional option to propagate the information of a node's features through its neighbours, but it still cannot match the performance of the $\mathcal{B}$ models.

It is quite standard in graph neural networks to have a direct dependence of $\boldsymbol{h}_u^{(t)}$ on $\boldsymbol{h}_u^{(t-1)}$, most notably in the Basic GNN Model [Hamilton, 2020], the Gated GNN Model [Li et al., 2017] and the GraphSAGE framework [Hamilton et al., 2017]. Where this dependency is not natively present, adding self-loops is a common and well-studied practice to improve performance. For example in the case of Graph Convolutional Networks, [Kipf and Welling, 2017] observe experimentally that adding self-loops improves accuracy on popular benchmarks and [Wu et al., 2019] provide theoretical support for these results. I considered doing this project about the effect of self-loops on the expressiveness of GATv2. However, I opted against this idea because [Brody et al., 2022] do use self-loops in some of their experiments and I preferred to examine a more innovative approach.

**(e)** The experimental results support the hypothesis that $\mathcal{B}$ outperforms $\mathcal{A}$ at capturing input-target correlation. However, it should be noted that $\mathcal{B}$'s benefit may not extend to more realistic datasets, as the presented task is unrealistic in at least two ways. First, it has been argued that the ER-model does not generate realistic graphs [Saberi, 2015]. The only reason why this model was chosen is for the sake of simplicity. Second, in the presented task there is no correlation between the graph stucture and the targets whatsoever. This situation never occurs in real-world graphs, as that would contradict the purpose of modelling the dataset as a graph.

Moreover, this study only considers models with one or two layers. While I hypothesize similar outcomes when using more layers, further study is required to assert this more confidently.

**(f)** The results in this study are not too surprising as the experiment was explicitly designed to test the aspect in which $\mathcal{B}$ surpasses $\mathcal{A}$. Therefore, it is imperative to compare the models on more realistic benchmarks to reach more conclusive statements. For this purpose, I suggest utilizing the *ogbn-products* benchmark [Hu et al., 2020] because of its information-rich input features.

Next, it is essential to compare the expressiveness of multi-layer models, both on the presented task and on more realistic benchmarks. This broader evaluation is necessary to extend the conclusion about the expressiveness of $\mathcal{A}$ and $\mathcal{B}$ beyond their one-layer and two-layer instances.

Finally, it would be useful to theoretically compare both architectures in the spirit of question 2d. It could be an intermediate step (and would be useful in its own right) to prove theoretical upper and lower bounds for $\mathcal{A}$ and $\mathcal{B}$ or simplications thereof. The 1-WL test is certainly an upper bound for both architectures as they are both MPNNs [Xu et al., 2018] and perhaps one can prove that this bound is also tight, like [Morris et al., 2021] did for the basic GNN model.

## References


[Brody et al., 2022] Brody, S., Alon, U., and Yahav, E. (2022). How attentive are graph attention networks?

[Erdős et al., 1960] Erdős, P., Rényi, A., et al. (1960). On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1):17–60.

[Hamilton et al., 2017] Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

[Hamilton, 2020] Hamilton, W. L. (2020). *Graph representation learning*. Morgan & Claypool Publishers.

[Hu et al., 2020] Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. (2020). Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133.

[Kipf and Welling, 2017] Kipf, T. N. and Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17.

[Leshno et al., 1993] Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867.

[Li et al., 2017] Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2017). Gated graph sequence neural networks.

[Morris et al., 2021] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. (2021). Weisfeiler and leman go neural: Higher-order graph neural networks.

[Rodgers and Nicewander, 1988] Rodgers, J. L. and Nicewander, W. A. (1988). Thirteen ways to look at the correlation coefficient. *American statistician*, pages 59–66.

[Saberi, 2015] Saberi, A. A. (2015). Recent advances in percolation theory and its applications. *Physics Reports*, 578:1–32.

[Wu et al., 2019] Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019). Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR.

[Xu et al., 2018] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018). How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.