Clément Apavou
Antoine Cadiou
Léo Tronchon
apavou.clement@outlook.fr
antoine.cadiou@ens-paris-saclay.fr
leo.tronchon@gmail.com

Practical Session # 3:
Graph-NN Homework
Deep Learning in Practice 2021

02/23/22

1 Diagram of the architecture

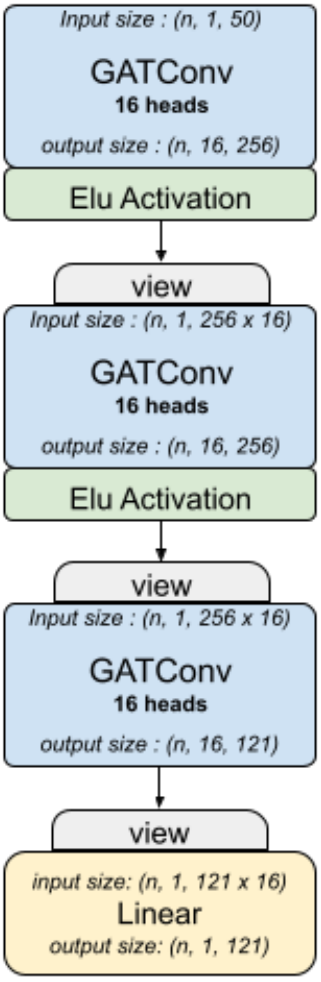


Figure 1: Rethinking Graph Attention Network (more heads is almost all you need) based on [1].

2 What kind of architectures did you try ?

2.1 Architecture

We used a model with multile GAT Conv layers as well as an aggregation/classification layer at the end of the GAT. Firstly, we reproduced the architecture of the original paper [1] and then we did experiments with a few architecture parameters to improve performances:

- We tried to modify the number of layers (1, 2, 3, 4) and ended up using 3.
- The number of heads for each GATConv layer (4, 8, 12, 16) and ended up using 16 heads. The difference in performance between 12 and 16 was minor however.
- The aggregation method. First we tried to use the last GATConv, by setting its number of heads to 1 and therefore we could obtain the number of outputs desired right away. Then we tried to aggregate it by taking the mean of the last GATConv (which was composed of multiple heads. Finally, we decided to let

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Figure 2: F1-score formula.

a Linear classification layer to reduce the dimension from $num\_heads * output\_size$ to $output\_size$. We obtained our best results with this last method.

2.2 Optimization

To train the model, we used the AdamW optimizer. We experimented with a few learning rates because they affected the training stability significantly. As can be seen on the training runs, the model's F1 score can drop dramatically, and using a high learning rate such as 0.005 leads to drops so important that the model does not recover. Therefore, after experimenting with $lr = 0.001, lr = 0.0008$, and $lr = 0.0005$ we ended up using $lr = 0.0008$ which yielded the best results. Since the large F1 score drops – although they get less important when decreasing the learning rate– never disappear completely, there is a trade-off in when decreasing the lr as very small learning rate takes longer to recover from each of them. Out of 1000 epochs $lr = 0.0008$ had the highest performance. Finally, still in an effort to regularize the training, we added a $weight\_decay = 0.04$.

3 What your result means ?

The F_1-score is the harmonic mean of precision and recall. It allows to evaluate the model on its precision and its recall. The $Recall$ allows to measure how well the model predicts a certain class and the $Precision$ measures how well the model is when it predicts a class. In our problem, each node has 121 labels and each label is a binary class associated to an attribute of the node. In our case, the class distribution could be imbalanced, so we used the F_1-score instead of the accuracy to take into account bad predictions of the minority classes properly, and fairly evaluate the performance of the model.

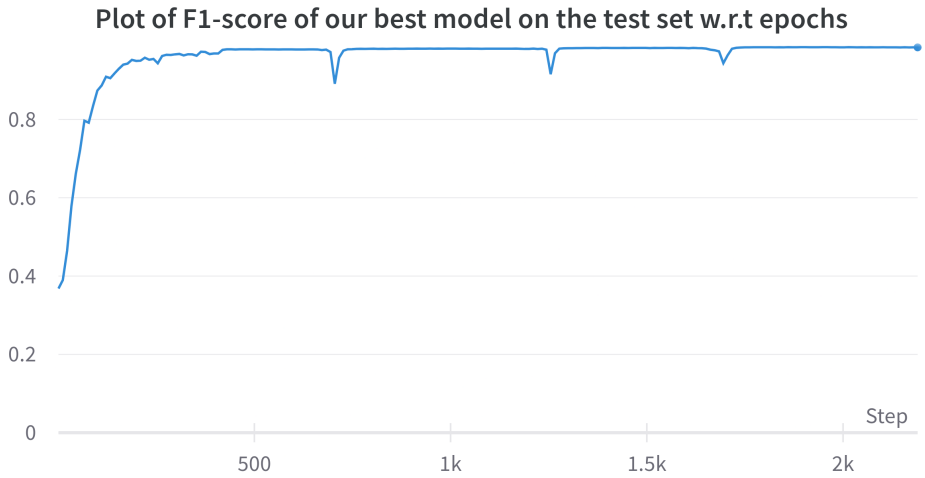


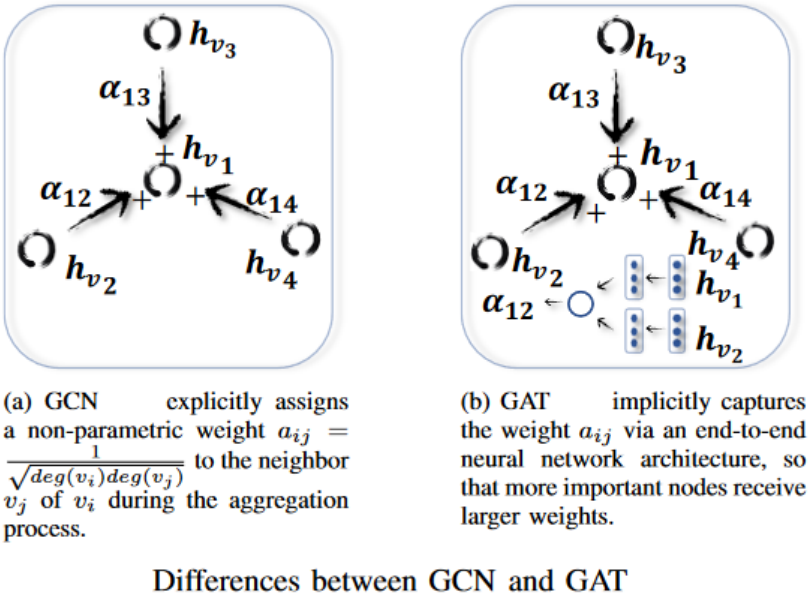
Figure 3: F1-score on the test set of our model w.r.t epochs (Run).

Architecture	Test F1-score	wandb Run
Original provided model	0.5053	None
GAT [1]	0.973	None
Our model 1	**0.9832**	Run

Table 1: Result of our model compared to different models on the PPI dataset.

4 Why would Attention Network perform better than GraphConv ?

Unlike GraphConv Networks, GAT [1] posits that contributions from adjacent nodes to the center node are not pre-determined (this difference is illustrated in Figure 4). To learn the relative weights between two linked nodes, GAT uses attention techniques. GAT also performs multi-head attention in order to improve the model's expressive capacity.
The GATConv-based model outperforms the GCN-based model because it does not depend on knowing the entire graph structure upfront contrarily to spectral-based approaches such as GCN. This is very important for inductive tasks such as this one, since we do not have a fixed graph to make node predictions on, but rather multiple graphs, which for the test set, the network has never seen before.



(a) GCN explicitly assigns a non-parametric weight $a_{ij} = \frac{1}{\sqrt{deg(v_i)deg(v_j)}}$ to the neighbor v_j of v_i during the aggregation process.

(b) GAT implicitly captures the weight a_{ij} via an end-to-end neural network architecture, so that more important nodes receive larger weights.

Differences between GCN and GAT

Figure 4: Differences between GCN and GAT (from this paper [2]).

5 Appendices

Weight & Biases all experiments : `https://wandb.ai/dlip_gnn/GNN-HW`
Github code : `https://github.com/clementapa/GNN-HW`

References

[1] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[2] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, Jan 2021.