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

Maxence Gélard
maxence.gelard@student-cs.fr

1. Architectures details of the best architecture

The goal of this assignment was to improve a neural network architecture for a classification task using the Protein-Protein Interaction (PPI) network dataset. Through different experiments detailed in section 2 of this report, I achieved to build a model using a Graph Attention Network (GAT), with the following architecture:

- Input layer: GAT layer ; input size 50 ; output size per attention head 256 ; 5 attention heads (so output size 1280 after concatenation), activation *LeakyRelu* (with a negative slope of 0.2) ; no residual connection,
- 4 Hidden layer: GAT layers ; input size 1280 ; output size 1280 ; 5 attention heads ; activation *LeakyRelu* (with a negative slope of 0.2) : residual connection
- Output layer: GAT layer ; input size 1280 ; output size 605 ; 5 attention heads ; no activation function (we use the logits for the cross entropy loss) ; residual connection. The output of size 605 is then average across the 5 attention heads to get a size of 121 (corresponding to the number of labels in the dataset

The big picture of this architecture, including the different inputs / outputs shapes, is given in Figure 1. For more clarity, the concatenation of the output of each attention head has been represented as a 3D tensor (e.g $(batchsize, 5, 256)$), but this is translated into a concatenation into a 2D tensor in the Python code $(batchsize, 1280)$. On this Figure, *GATLayer* includes the attention mechanism which is detailed in Figure 2. There, I have used the shape of the input layer, but this translates in the same way with the other layers' shape. Note that on the figure $a_{i,j}^L$ is a learnable parameter of the edge attention. For this Figure 2, I got helped by the official documentation of the DGL library [1].

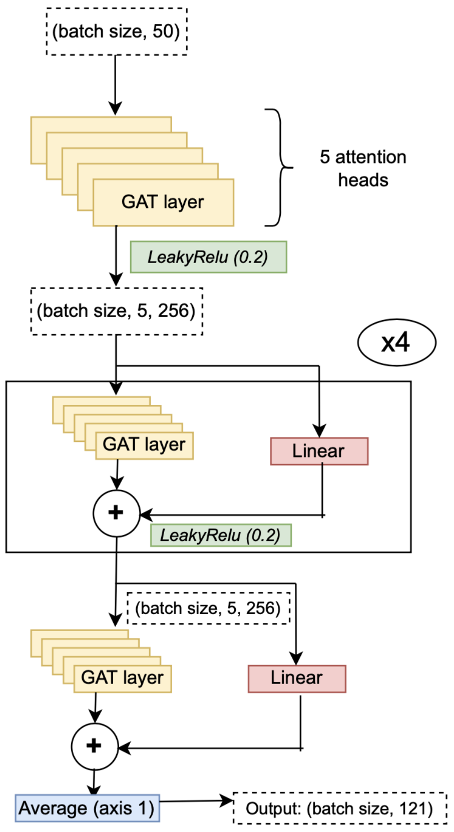

Figure 1. Best model architecture

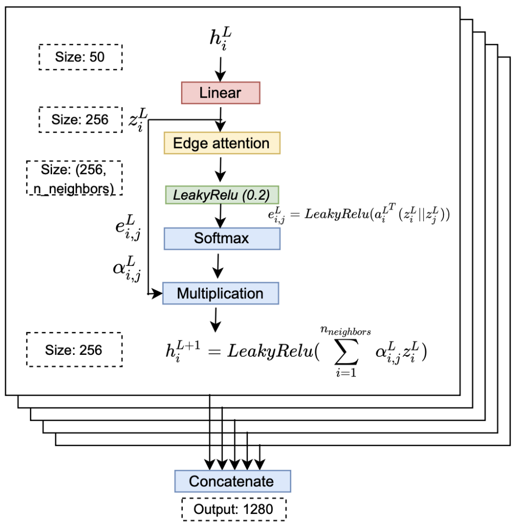

Figure 2. GAT attention mechanism

2. Results and discussion

In this section, we will be discussing the different experiments I have made, covering architecture choice and parameters tuning, as well as give some understanding of the role of Attention mechanism to achieve better results. For this part, I found some help in the Deep Graph Library documentation [1], as well as some insight from the Graph Attention Network paper [2].

2.1. Architecture and parameters tuning

2.1.1 Metric: micro F1 score

In order to select the best architecture, as well as to tune its parameters, we need to define the metric that we will be using in order to measure the performance of a given model. We are in our usecase in a multi-label classification task. We will be using the *F1 score*, using *micro-averaging*. First we need to define what are precision and recall, keeping in mind that in contrary to the binary case, these quantity will be computed per class:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$
$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

Then, we can compute a *per class F1 score*: $2\frac{Precision \times Recall}{Precision + Recall}$. Similarly, we define micro F1 score as:

$$microF1 = \frac{MicroPrecision \times MicroRecall}{MicroPrecision + MicroRecall}$$

where $MicroPrecision$ and $MicroRecall$, corresponds to $Precision$ and $Recall$, but instead of considering one class vs. all others, we take into account the total number of True Positive / False Positive / False Negative.

One can observe that overall, the *micro F1 score* boils down to be the proportion of correctly classified examples out of the total number of examples, i.e this is the model's **accuracy**.

2.1.2 Experiments

For this assignment, I have been trying 3 different models: Graph Convolutional Network, Linear Network (i.e with Dense layer), Graph Attention Network (GAT).

As my best architecture, mentioned in part 1, turned out to be a GAT Network with 5 layers, I first carried out a comparison experiment of these three architectures, keeping the same number of hidden layers for all of them (with 256 as hidden dimension). I also compared these 3 architecture with the baseline model which was a Graph Convolutional Network with 2 hidden layers. Below are given the results:

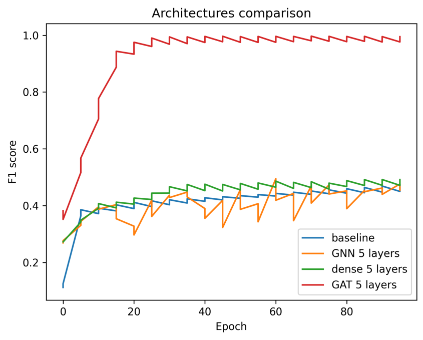

Figure 3. Architectures comparison

One can see that GAT network does provide the best result, with a F1-score close to 1, and the two other remaining at around 0.5. As it will be discussed in the last sub-section, it means that the architecture of Graph Convolutional Network (and even more for Linear models) is not expressive enough to provide correct classification results. Moreover, one can also observe that increasing the number of layers (2 to 5 here) for the classic Graph Convolutional Network doesn't help improving the performance (and even made the learning curve more oscillating), meaning that the issue was the network architecture and not really the tuning of its parameters.

I was also interested in understanding the influence of different parameters on the GAT architecture, especially the number of layers, the number of attention heads in each layer as well as the use of residual connections. Below are given the results of different experiments on these parameters (when one parameter varies, the other are set to their best values, given in section 1):

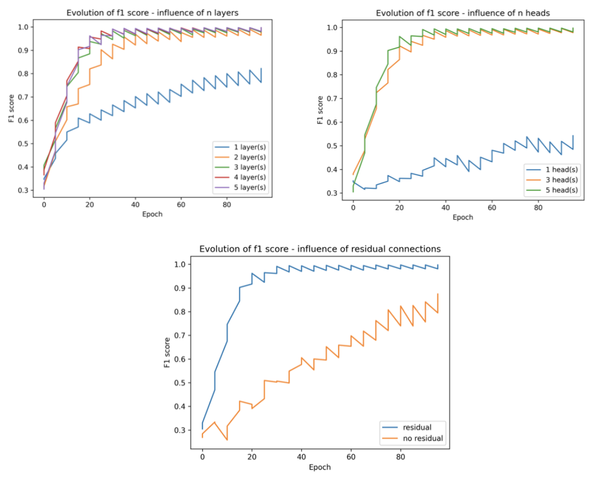

Figure 4. Influence of hyperparameters on GAT F1-score

First, one can see that increasing the number of layers and the number of attention heads allows the model to perform better (in terms of F1-score). Moreover, using residual connection incredibly boosted the performance of the model, as well as smoothed the training process (less oscillation of the F1 curve). Therefore, my best architecture has been deduced from these experiment: 5 hidden layers, 5 attentions heads and residual connection between hidden layers.

2.2. *GraphConv* vs *Graph Attention Network*

In this last part, we will give some intuition on why would Attention Network perform better than GraphConv. In a Graph Convolutional Network, each layer computes activations for nodes and edges based on previous values of activations of neighbors. Therefore, such architecture is really structure dependant, and this prior on the structure may result in poor generalizability of the network, hence bad performance. On the other hand, Graph Attention Network allows for nodes to attend to all of their neighbors' features and thus lead to a different aggregation of activations, without sticking to the pre-defined underlying graph structure, thus leveraging the model's capacity. Moreover, attention's mechanism can be extended to the case of multi-head attentions to even more enrich the model which also makes the learning process more stable. This is indeed what we observed in the previous experiments, with the GAT model achieving almost twice the performance of the classic GNN. We also notice that adding residual connections between hidden layers made the training even more easier when considering deeper models (here with 5 hidden layers) by feeding the previous layer's input directly to the next GAT layer.

References

[1] DGL Team. Deep graph library tutorials and documentation. *Available at* `https://docs.dgl.ai/index.html`*, visited on 07.02.2022*.

[2] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.