Effects of Graph Pooling Layers on Classification with Graph Neural Networks

Linda Studer[*†], **Jannis Wallau**[*], **Rolf Ingold**[*], **and Andreas Fischer**[*†]

[*]*Document Image and Video Analysis Group (DIVA)*
University of Fribourg, Switzerland
{firstname}.{lastname}@unifr.ch

[†]*Institute of Complex Systems (iCoSys)*
University of Applied Sciences and Arts Western Switzerland
{firstname}.{lastname}@hefr.ch

***Abstract*—With the rise of graph neural networks, sometimes also referred to as geometric deep learning, a range of new types of network layers have been introduced. Since this is a very recent development, the design of new architectures relies a lot on intuition and trial-and-error. In this paper, we evaluate the effect of adding graph pooling layers to a network, which down-sample graphs, and evaluate the performance on three different datasets. We find that especially for smaller graphs, adding pooling layers should be done with caution, as they can have a negative effect on the overall performance.**

***Index Terms*—graph neural networks, graph pooling, graphs**

I. Introduction

A very recent development in the field of deep learning is the extension of convolution from the Euclidean space, such as images and videos, to the non-Euclidean space, such as graphs [1]. Graphs consist of nodes and edges, which connect the nodes. Nodes as well as edges can have additional information attached, typically in form of real-valued features. Like in the Euclidean space, graph convolution aggregates the information of the local neighbourhood. In the graphs, the neighbourhood of a node is defined by the edges that connect it to other nodes. Based on the features of the neighbouring nodes, and sometimes also the features of the connecting edge, graph convolution computes a new hidden representation for each node. For example, a very simple aggregation function is taking the average of the features. Another type of layers that have the potential to improve a graph neural networks (GNN) are graph-pooling layers [2]. Much like pooling layers in Convolutional Neural Networks (CNNs) are used to down-sample images, these layers are able to down-sample graphs. They compute a pooled graph (X', A') from an input graph (X, A), where X is the matrix of node features and A is the adjacency matrix [2].

This new part of the deep learning field has not only opened up a whole new range of methods but also brought challenges. One of these challenges is related to the network depth, i.e. the number of layers. It has been demonstrated that going deeper, which is often able to improve the performance and allows for a more diverse set of filters to be learned in the Euclidean space, does not necessarily improve convolutional GNNs, but rather has the opposite effect [1]. Similarly, in this paper, we aim to investigate the effect of adding graph-pooling layers on the classification performance of GNNs.

The work presented here has been partially supported by the Rising Tide foundation with the grant number CCR-18-130.

TABLE I
Comparison between the three datasets.

Dataset	# Classes	# Nodes per graph		
		min	max	median
Letter (LOW)	15	1	8	5
AIDS	2	2	95	11
pT1-Gland-Graph	2	16	639	65

II. Study Design

In our experimental study, three different architectures with a different number of graph pooling layers are trained. In order to evaluate the effect of graph pooling, we select three datasets from diverse domains with different graph sizes. Each dataset has node features available and is annotated for graph classification.

A. Datasets

The graphs sizes are listed in Table I. The Letter (LOW) dataset contains the smallest graphs, the pT1-Gland-Graph dataset the largest.

1) Letter Dataset (LOW): The graphs in this dataset represent distorted letter drawings of 15 capital letters (A, E, F, H, I, K, L, M, N, T, V, W, X, Y, Z) [3]. Three levels of distortion have been applied to the base dataset, we use the subset with the lowest distortion. The nodes are labelled with the x and y coordinates. The training, validation, and test set each contains 750 graphs.

2) AIDS Database: The AIDS database [3] consists of 4,337 graphs, of which 2,337 are used as the test set. Each graph represents a molecule that is either active or inactive against HIV. Each atom in the molecule is represented as a node and each bond between the atoms is represented as an

978-1-7281-7177-7/20/$31.00 ©2020 IEEE
DOI 10.1109/SDS49233.2020.00021

TABLE II
ARCHITECTURES USED FOR THE EXPERIMENTAL EVALUATION. THEY ALL HAVE A DIFFERENT NUMBER OF GRAPH POOLING LAYERS, BUT THE SAME READOUT AND LINEAR LAYERS. TKP STANDS FOR TOP-k POOLING.

ARCHITECTURES		
0-TKP	1-TKP	3-TKP
GCNCONV GCNCONV GCNCONV	GCNCONV GCNCONV TOPKPOOLING GCNCONV	GCNCONV TOPKPOOLING GCNCONV TOPKPOOLING GCNCONV TOPKPOOLING
GLOBAL MEAN-POOLING LINEAR LINEAR (CLASSIFICATION)		

edge. Each node is labelled with the symbol of the chemical element (e.g. C for carbon), its charge and the coordinates.

3) pT1 Gland Graph Dataset: The pT1-GG dataset [4] consist of 520 graphs, 130 of which are used as the test set. These graphs have been extracted from images of pathological tissue sections from colorectal cancer patients. Each graph represents of either a normal or dysplastic intestinal gland. Each cell of a gland is represented as a node, and each node is connected to its spatially two closest neighbours. 33 features are available for each node, such as staining intensity and the size of the cell.

B. Experimental Setup

We use the three different architectures with a different number of pooling layers (0, 1 and 3). Table II gives an overview of the architectures. For the graph convolution layers we use Graph Convolutional Network (GCN) [5] layers with 128 neurons. GCN layers can only use the node features and do not consider edge features. For the graph pooling we use the top-k pooling layer [2]. It is called top-k because it reduces the number of nodes in a graph from N to kN after the pooling layer, where $k \in (0, 1]$. We use a k of 0.8. In order to get a graph-level output, all the node features are averaged, so that for a single graph $\mathcal{G}_i$ its output is computed as $\mathbf{r}_i = \frac{1}{N_i} \sum_{n=1}^{N_i} \mathbf{x}_n$, where $\mathbf{x}_n$ is the feature vector of a node. To put our classification results into context, we also evaluated the performance using the GCN architecture from published work [5] with 3 layers and 128 neurons per layer. The setup is the same as the architecture without pooling layers, but with an added dropout layer between the two linear layers. All experiments are run using the PyTorch Geometric library [6] and the implementation details as well as the hyper-parameters are available open-source[1].

Each experiment is run 10 times until convergence and the mean and standard deviation of the achieved accuracy are reported. Z-normalisation is used to adjusts each node feature value x such that $\hat{x} = \frac{x-\mu}{\sigma}$. For the AIDS database and the Letter (LOW) dataset, we use the same dataset split as published [3]. For the pT1-GG dataset, we use the first cross-validation split as published [4].

[1]https://github.com/LindaSt/SDS-2020

TABLE III
CLASSIFICATION RESULTS (IN %) FOR THE THREE DATASETS USING THE FOUR DIFFERENT ARCHITECTURES (MEAN AND STD OVER 10 RUNS).

ARCHITECTURE	LETTER (LOW)	AIDS	PT1-GG
0-TKP	81.8 ± 0.8	93.2 ± 3.9	93.5 ± 2.2
1-TKP	79.7 ± 2.0	90.3 ± 6.0	94.2 ± 1.0
3-TKP	78.7 ± 1.7	88.5 ± 6.6	90.8 ± 3.8
GCN [5]	82.9 ± 0.7	94.8 ± 2.1	94.5 ± 2.8

III. RESULTS

Table III provides an overview of the achieved performances. Adding three pooling layers has a negative effect on the average performance on all three datasets. However, for the pT1-GG dataset, adding just one pooling layer slightly improves the performance. For the other two datasets, the model without pooling layers performs the best.

IV. CONCLUSION

We find that adding top-k pooling layers generally has a negative effect on the performance. Only the pT1-GG dataset with the largest graphs profits from a pooling layer, which suggests that the usefulness of graph pooling is linked to the graph size. This makes sense intuitively, as removing nodes from a small graph has a much larger impact on the overall structure. However, with just these finding, it remains difficult to fully establish the cause of the drop in performance.

Overall, our findings support previous work which states that adding more layers, i.e. more learnable parameters does not necessarily improve the performance [1]. Furthermore, although graph pooling can be beneficial for certain graph types, one should be careful when dealing with smaller graphs. It seems that experiences from image-based deep learning are therefore not directly applicable in geometric deep learning.

In the future, to make a more in-depth analysis of this phenomena, the pooled graph representations should be examined to investigate exactly which nodes in the graphs are combined. Other aspects could also play a role, such as the graph connectivity.

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.

[2] C. Cangea, P. Veličković, N. Jovanović, T. Kipf, and P. Liò, "Towards sparse hierarchical graph classifiers," *arXiv preprint arXiv:1811.01287*, 2018.

[3] K. Riesen and H. Bunke, "Iam graph database repository for graph based pattern recognition and machine learning," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition and Structural and Syntactic Pattern Recognition*. Springer, 2008, pp. 287–297.

[4] L. Studer, S. Toneyan, I. Zlobec, H. Dawson, and A. Fischer, "Graph-based classification of intestinal glands in colorectal cancer tissue images," in *2nd COMPAY Workshop at MICCAI*, 2019.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[6] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.