Spatial Information in Graph Convolutional Neural Networks

Bartłomiej Wójcik and Arkadiusz Tomczyk[0000−0001−9840−6209]

Institute of Information Technology
Lodz University of Technology
al. Politechniki 8, 93-590 Lodz, Poland
arkadiusz.tomczyk@p.lodz.pl

Abstract. In recent years, to enable application of convolutional neural networks to structures other than regular grids (e.g. graphs), different definitions of convolution operation were proposed. Some of them, however, do not take into account spatial coordinates of graph nodes, if they are available in the considered dataset, and focus only on node features. In this work we propose three strategies, which allow to add that information to network architectures. The presented approach leads to results improvement on four benchmark datasets.

Keywords: Convolution · Graph neural networks · Spatial information.

1 Introduction

Convolutional neural networks CNN ([7]) are still state-of-the-art methods of image processing. Analyzing working principle of single convolutional layer[1], it is evident that such classic convolution inherently depends on spatial distribution of pixels. To be more specific, calculating result $\mathbf{h}'_i \in \mathbb{R}^m$ of single convolution layer for pixel i we consider features $\mathbf{h}_j \in \mathbb{R}^n$ of pixels j in the local neighborhood $\mathcal{N}(i)$. The relative position $\mathbf{p}_{j,i} = \mathbf{p}_j - \mathbf{p}_i \in \mathbb{R}^d$ allows to identify corresponding parameter of trainable kernel (mask).

Currently, convolutional neural networks are applied not only to regular structures (grids) but also to irregular ones (graphs). Graph nodes are also described with feature vectors, as pixels in the case of images. There are however substantial differences. First of all, the neighborhood $\mathcal{N}(i)$ is not fixed - different nodes can have different number of neighbors. Secondly, the spatial relations between nodes need not be known since in general case nodes need not to have spatial coordinates assigned (e.g. in social networks) and even if they have, those coordinates need not to be expressed by integer numbers. Finally, additional knowledge can be encoded in features of edges connecting graph nodes[2]. Consequently, the definition of convolution operation had to be modified to solve

[1] In this paper we focus only on convolutions and additional elements of CNNs achitectures like: non-linearities, normalizations, etc., are omitted.

[2] Edge features, although considered in many graph convolutional networks, are not considered in this work to focus only on spatial coordinates of nodes.

all of those problems. Most of the existing approaches ([6, 4, 8, 5, 1, 2]), can be described in the following form:

$$\mathbf{h}'_i = \mathrm{LIN}_{\mathbf{L}(i),\mathbf{l}(i)}(\mathbf{h}_i) + \square_{j \in \mathcal{N}(i)} \mathrm{LIN}_{\mathbf{M}(j,i),\mathbf{m}(j,i)}(\mathbf{h}_j) \tag{1}$$

Matrices $\mathbf{L}(i)$, $\mathbf{M}(j,i) \in \mathbb{R}^{m \times n}$ and vectors $\mathbf{l}(i)$, $\mathbf{m}(j,i) \in \mathbb{R}^m$ can be fixed or contain trainable parameters, $\square$ denotes a differentiable, permutation invariant operator, e.g. sum, mean, max, etc., whereas LIN[3] is a linear transformation of feature vectors. Some of those approaches in a natural way can handle spatial coordinates of graph nodes. The rest of them, however, do not consider it all. In this work we investigate how to use effectively the latter architectures in problems where node positions are available.

2 Method

Classic graph convolutional neural networks handle node positions in three typical ways. Firstly, some of them, e.g. MoNet [8] and SplineCNN [4], try to naturally generalize classic CNNs. It requires training of continuous kernel functions (masks) to directly operate on continous relative coordinates $\mathbf{p}_{j,i}$. They model them using either gaussian mixture model or B-spline bases, respectively. Secondly, to our best knowledge, there is one method SGCN [2] which tries to tackle the same problem without following CNN strategy. In this case:

$$\mathbf{M}(j,i) = \mathrm{diag}(\sigma(\mathbf{U}\mathbf{p}_{j,i} + \mathbf{b})) \tag{2}$$

where $\mathbf{U} \in \mathbb{R}^{n \times d}$, $\mathbf{b} \in \mathbb{R}^n$ and σ is a nonlinear function (ReLU was used). A limitation of such formulation is the fact that here $m = n$. To partially avoid it, results of several operations can be concatenated. Finally, there is a numerous group of methods like: GCN [6], GAT [1], GraphSAGE [5] etc., that simply ignore those positions focusing on node features only.

The most straightforward option to take into account node coordinates $\mathbf{p}_i$ in the latter models would be adding them (concatenation) to the feature vector $\mathbf{h}_i$. This approach alone leads to the results enhancement but there is still space for further improvement. That is why in this work we propose three strategies allowing to inject additional spatial information into convolutional operation:

- $\mathbf{M}(j,i) \rightarrow \mathbf{M}(j,i) \cdot \mathbf{a}^T \mathbf{p}_{i,j}$ - linear strategy where $\mathbf{a} \in \mathbb{R}^d$ is an additional trainable parameter
- $\mathbf{M}(j,i) \rightarrow \mathbf{M}(j,i) \cdot \mathrm{MLP}_{\mathbf{A_2},\mathbf{b_2},\mathbf{A_1},\mathbf{b_1}}(\mathbf{p}_{i,j})$ - non-linear strategy where MLP[4] is a multilayer perceptron with trainable parameters $\mathbf{A_2} \in \mathbb{R}^{1 \times k}$, $\mathbf{b_2} \in \mathbb{R}$, $\mathbf{A_1} \in \mathbb{R}^{k \times d}$ and $\mathbf{b_1} \in \mathbb{R}^k$ (k denotes the number of hidden units, there is 1 output)

[3] $\mathrm{LIN}_{\mathbf{A},\mathbf{b}}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$

[4] $\mathrm{MLP}_{\mathbf{A_2},\mathbf{b_2},\mathbf{A_1},\mathbf{b_1}}(\mathbf{x}) = \mathrm{LIN}_{\mathbf{A_2},\mathbf{b_2}}(\sigma(\mathrm{LIN}_{\mathbf{A_1},\mathbf{b_1}}(\mathbf{x})))$

Table 1: Comparison of classic approaches and proposed strategies applied in GraphSAGE. All results were obtained using PyG framework (average value from 5 runs is presented). Some of them, for superpixel-based representation of MNIST available in PyG, were taken from [2] (in particular result for SGCN method).

Method	MNIST	MNIST PyG	Fashion MNIST	CIFAR10	AIDS
without position in input features					
SplineConv	97.21%	95.22% [2]	87.46%	49.53%	81.47%
GMMConv	96.84%	91.11% [2]	86.62%	45.68%	80.58%
GraphSAGE	94.15%	79.88%	84.72%	55.69%	79.73%
SGCN	-	95.95% [2]	-	-	-
Ours linear	96.72%	97.49%	87.37%	63.03%	**95.62%**
Ours non-linear	98.29%	97.01%	87.22%	65.52%	95.58%
Ours enhanced	**98.90%**	**97.91%**	**89.47%**	**67.68%**	95.09%
with position in input features					
SplineConv	97.52%	97.83%	87.50%	63.03%	95.62%
GMMConv	97.07%	96.10%	87.22%	61.80%	93.08%
GraphSAGE	98.01%	97.13%	88.09%	66.32%	93.48%
Ours linear	98.45%	97.78%	88.35%	65.11%	**97.59%**
Ours non-linear	99.05%	98.32%	88.41%	67.07%	96.96%
Ours enhanced	**99.10%**	**98.81%**	**90.98%**	**73.23%**	96.65%

– $\mathbf{M}(j,i) \rightarrow \mathbf{M}(j,i) \cdot \mathrm{diag}(\mathrm{MLP}_{\mathbf{A_2},\mathbf{b_2},\mathbf{A_1},\mathbf{b_1}}(\mathbf{p}_{i,j}))$ - enhanced non-linear strategy, here $\mathbf{A_2} \in \mathbb{R}^{n \times k}$, $\mathbf{b_2} \in \mathbb{R}^n$, $\mathbf{A_1} \in \mathbb{R}^{k \times d}$ and $\mathbf{b_1} \in \mathbb{R}^k$ (there are n outputs)

It is worth noticing that this is a generic approach. It can be used with any convolution operation that can be described by formula (1).

3 Results

In the conducted experiments we have analyzed how the proposed strategies influence the results of GraphSAGE model where $\mathbf{L}(i) = \mathbf{W}_1$, $\mathbf{M}(j,i) = \mathbf{W}_2$ are trainable matrices, $\mathbf{l}(i) = \mathbf{0}$, $\mathbf{m}(j,i) = \mathbf{0}$ and $\square$ is a mean operator. Table 1 presents the results of those experiments with coordinates $\mathbf{p}_i$ included in node feature vectors $\mathbf{h}_i$ of input signal and without them, respectively, for four different datasets. First three of them are superpixel-based representations of images (MNIST, Fashion MNIST, CIFAR10), whereas the last one (AIDS) contains graph representation of chemical molecules. To make the results comparable in all cases three convolutional layers were considered with ReLU as a non-linear activation function. Since classification task was considered convolutional layers were followed by MLP (two fully-connected layers with ReLU between them)

and while training cross-entropy lass was used. In all MLPs there were 64 hidden units.

Our results successfully demonstrate the increase of performance achieved across all four datasets. In the case of image datasets, the enhanced non-linear method was the most effective. Chemical compound AIDS dataset obtained the highest accuracy using linear strategy. Conducted experiments show also evident improvement when additionally coordinates $\mathbf{p}_i$ are included in node feature vectors $\mathbf{h}_i$. Since for the same image different superpixel-based representations can be generated, to compare results with SGCN we have performed additional experiments for MNIST dataset available in PyG framework ([3]). Also in this case our strategies lead to better results.

4 Summary

In this work we have shown that properly used spatial information can improve outcomes of graph convolutional neural networks, even if in its original form they do not consider nodes coordinates at all. We have proposed several strategies and proved (using GraphSAGE model and four well-known benchmarks) their positive influence on the classification results. Our further research will focus on experimenting with other models and datasets where spatial information is available. Moreover, we want to check other strategies as well as apply this approach for other tasks (e.g. node classification).

References

1. Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks? (2021)
2. Danel, T., Spurek, P., Tabor, J., Smieja, M., Struski, L., Slowik, A., Maziarka, L.: Spatial graph convolutional networks. In: Neural information processing 27th International Conference, ICONIP 2020 : Bangkok, Thailand, November 18-22, 2020 : proceedings, part V, pp. 668–675. Communications in Computer and Information Science, ISSN 1865-0929, eISSN 1865-0937; Vol. 1333, Springer, Cham (2020)
3. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)
4. Fey, M., Lenssen, J.E., Weichert, F., Müller, H.: SplineCNN: Fast geometric deep learning with continuous b-spline kernels. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 869–877 (2018)
5. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. p. 1025–1035. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)
6. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017)
7. Lecun, Y., Bengio, Y.: Convolutional Networks for Images, Speech and Time Series, pp. 255–258. The MIT Press (1995)
8. Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5425–5434. IEEE Computer Society, Los Alamitos, CA, USA (jul 2017)