







Contextual Image Classification Through Fine-Tuned Graph Neural Networks

Walacy S. Campos^(✉) , Luis G. Souza , Priscila T. M. Saito ,
and Pedro H. Bugatti 

Department of Computing, Federal University of Technology - Parana,
1640 Alberto Carazzai Ave, Cornelio Procopio, Parana, Brazil
{walacycampos, souza.1998}@alunos.utfpr.edu.br,
{psaito, pbugatti}@utfpr.edu.br

Abstract. Nowadays, computer vision techniques have become popular in several domains (e.g., agriculture, industry, medicine, and others). Their success derives from the advances in computational resources and the large volume of complex data (i.e., images). These factors led to an increase in the use of convolutional neural networks. However, such deep learning architectures do not appropriately explore the relationships between the data (e.g., images) and their respective structure. To better gather and encodes these affinity connections into a deep neural network, we can use the so-called graph neural networks (GNNs). These graph-based networks also present drawbacks. The high number of relationships in a graph can be considered a bottleneck regarding the available resources and scalability. Hence, to mitigate this issue, we propose to use GNNs automatically tuned, defining their well-suited connections according to a given image context, which improves their efficiency and efficacy. We performed experiments considering different types of state-of-the-art deep features aggregated with the GNNs. The results demonstrate that our proposed method can achieve equal accuracy (statistically) to GNNs with complete and random connections. Moreover, we decreased the number of edges to a great extent (up to 96%), testifying to our method's effectiveness.

Keywords: Deep learning · Convolutional neural network · Graph neural network · Graph pruning · Computer vision

1 Introduction

Computer vision has been used widely in different contexts like agriculture, medicine, industrial, etc. Part of this success is due to the massive volume of complex data generated daily by these domains. Moreover, with advances in computational resources (e.g., GPUs), it was possible to process (e.g., classify) all these data in an efficient way [2].

A powerful computer vision technique that was boosted by the recent advances was the so-called Convolutional Neural Network (CNN). Although CNNs present good results, they are incapable of gathering and harnessing the relationship between the data. These bonds between the data can considerably improve the effectiveness of the

classification process. Then, to reach this important property, it is possible to use GNNs [21] aggregated with CNNs.

As well-known, a graph structure can be highly affected by factors like the number of nodes and edges. This behavior leads to a significant drawback when using GNNs to image classification. The higher the number of edges, the higher the memory footprint and the cost to generate a suitable learning model. The cost of training a GNN is proportional to the number of edges. Moreover, we cannot generate a random or complete graph regarding image classification since it neglects semantic connections and impacts the accuracy of the model. A typical approach to solve this problem and build an appropriate graph is the brute force policy (e.g., grid-search). The problem with this strategy is a high computational cost. Some works tried to consider a priori graph (e.g., knowledge graph) to specific domains to solve this issue [4, 5, 14] but, unfortunately, in dynamic and real scenarios is almost impossible to obtain such a priori structure.

Thus, to mitigate these drawbacks, in this paper, we propose to use GNNs automatically tuned, defining their well-suited connections according to a given image context, improving their efficiency and efficacy. To do that, we based our approach according to the similarity between the graph nodes (i.e., images). Despite its simplicity, our approach achieved notable results. It significantly reduced memory footprint when using GNNs aggregated with CNNs and obtained good accuracy compared with random and complete graphs. Also, we achieve better representativeness of the semantic between an image and its objects to define the global classification context. It is worth mentioning that the literature works [6, 10, 13] tried to apply similar approaches, however disregarding GNNs aggregated with CNNs to define the global image context.

In summary, our main contributions are: i) we proposed an approach capable of tuning and reaching a well-suited GNN structure to improve the effectiveness of image classification context; ii) our approach also diminishes the computational cost of GNNs aggregated with CNNs; iii) through our tuning policy we provide greater structural semantics to the learning model; iv) since our tuning process is based on similarity, our approach can be straightforwardly extended using several literature practices that use the same concept (e.g., different distances, indexes, pruning policies, among others).

2 Background

In [18] the authors presented the seminal work regarding GNNs. They discussed issues about the traditional neural networks handling information between their features. Their work has guided many other GNNs approaches [7, 8, 16]. Indeed, GNNs were motivated by CNNs due to their ability to extract spatial features in many scales and build expressive representations [22]. In [3] the authors introduced convolutional filter usage in GNNs, generalizing the concept of CNNs applied to Euclidean spaces to non-Euclidean ones. Equation 1 formally defines the convolution operation on graphs.

$$g_{\theta \star} \approx \sum_{k=0}^K \theta' T_k(L_{sym}) s \quad (1)$$

where $s \in R^n$ is a graph signal, g_{θ} is a spectral filter, the symbol \star is the convolutional operator, T_k refers to the Chebyshev polynomials, $\theta' \in R^K$ is a vector of Chebyshev

coefficients, and L_{sym} represents the normalized Laplacian graph $L_{sym} := D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$, considering the value for Laplacian Graph as $L := D - A$, $A = [a_{ij}]$ as the adjacency matrix non-negative and $D = \text{diag}(d_1, d_2, \dots, d_n)$ as the degree matrix of A where $d_i = \sum_j a_{ij}$ is the degree of vertex i [13].

In [12] it was proposed a graph convolutional network (GCN) as defined by Eq. 2. The authors simplified the model limiting nearest neighbors value in the convolution to $k = 1$, and approximating the largest eigenvalue (λ_{max}) of L_{sym} by 2.

$$g_{\theta} \star s = \theta(I + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})s, \quad (2)$$

where θ is considered the remained Chebyshev coefficient. The next process consists in applying a normalization trick to the convolution matrix (see Eq. 3):

$$I + D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}, \quad (3)$$

where $\tilde{A} = A + I$ and $\tilde{D} = \sum_j \tilde{A}_{ij}$.

Finally, the GCN itself can be described in Eq. 4.

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}\Theta^{(l)}) \quad (4)$$

where $H^{(l)}$ is the matrix of activations for the l -th layer, while $H^{(0)} = X$, $\Theta^{(l)} \in R^{a \times f}$ is considered as the trainable weight matrix in the layer l , and σ the activation function (i.e., $ReLU(\cdot) = \max(0, \cdot)$) [13].

Although very promising, the work of [12] presents some issues regarding the memory footprint, and it is applied to text classification (i.e., the graph structure is intrinsically defined). Moreover, it assumes that the edges have equal relevance to generate the learning model. Hence, it neglects that in real scenarios, each connection can show a different contribution. Besides, in [13] it is described that GCNs lose their representation power when several convolutional layers are added to the architecture, and the performance complexity is prohibitive.

We believe that our approach can cope with these issues since we consider different relevant levels to edges according to the image context. Besides, through this relevance, we can remove useless edges (i.e., not contribute to gather and harness the semantic relationship between an image and its objects). In [6] the authors proposed a technique based on sharing network information to define the importance of a given edge in a tree structure regarding traditional data. However, to the best of our knowledge, our work is the first to consider a relevance mechanism in GNNs aggregated with CNNs to define the global context of images (complex data).

3 Proposed Approach

To define the well-suited connections for the GNNs according to a given image context, we present the proposed approach pipeline, as illustrated in Fig. 1. In the first step, we obtain the objects (bounding boxes) from the images of the dataset. After that, in Step 2, features of the bounding boxes are extracted through pre-trained CNNs (e.g., ImageNet transfer learning). In Step 3 the bounding boxes belonging to the same image become nodes in a complete subgraph.

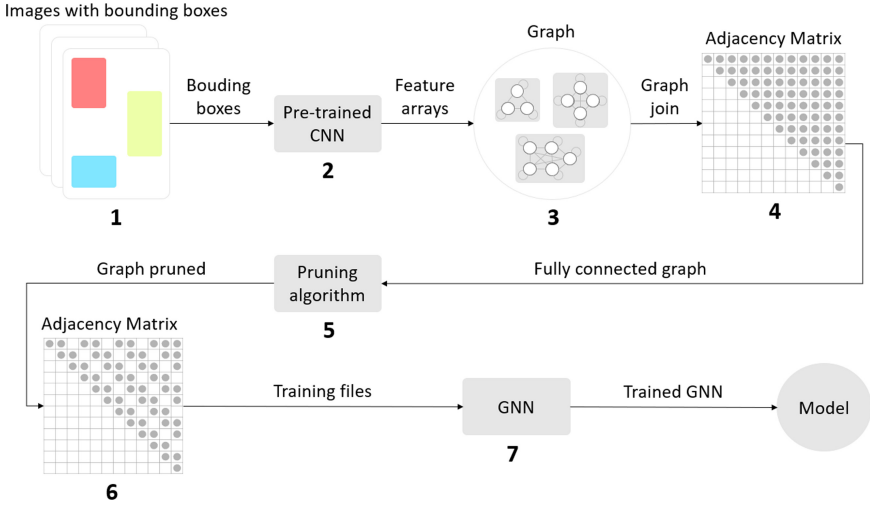


Fig. 1. Pipeline of the proposed approach.

Next, for each image is built a subgraph $G(V, E)$, where V represents the bounding boxes of the images and E their relations. Each bounding box from a given image is connected with their respective siblings (i.e., remaining bounding boxes from the same image). Then, in Step 4, an affinity matrix is generated to describe the entire graph (composed of several subgraphs). It is easy to see that the entire graph comprises several subgraphs, and the number of subgraphs is equal to the number of images in the dataset.

In Step 5, we automatically fine-tune the generated graph, defining their well-suited connections according to a given image context. To do so, we compute the relevance of each vertex according to the mean distance regarding its incident vertices. It is worth mentioning that we assign for each edge a relevance based on the dissimilarity between its nodes. The vertex's relevance is used as a threshold (τ) to suppress useless incident edges. Equation 5 formally defines the threshold setting. Figure 2 illustrates an example of the complete graph (left graph), and it is respectively fine-tuned one (right), obtained after applying our proposed approach suppressing edges according to the automatic threshold.

$$\tau = \frac{\sum_{k=0}^{l-1} w(n'_k, n''_k)}{l} \quad (5)$$

where l represents the number of edges, k is an iterator, $r(v'_k, v''_k)$ is a function to obtain the relevance of the edge connecting vertices v'_k and v''_k .

After, in Step 6, our approach creates the affinity matrix of the fine-tuned graph. Finally, in Step 7, this matrix is the input to train a GNN, generating a learning model.

In this paper, we create an instance of our proposed approach to calculate the edge's relevance through the well-known Euclidean distance. However, our approach can be straightforwardly extended to different kinds of distance functions, as well as tuning

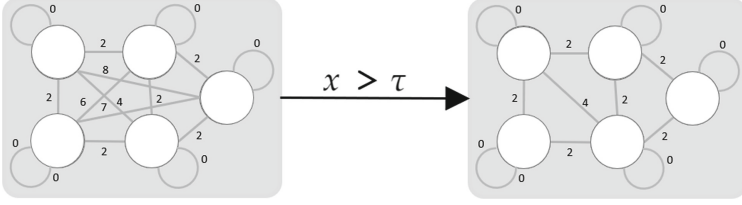


Fig. 2. Illustration of the tuning process in a fully connected graph

mechanisms. Algorithms 1 and 2 detail our proposed approach and the tuning mechanism considered in the present paper.

An important property of our proposed approach is that we do not need to know the bounding boxes labels to predict the global context of the image. As aforementioned in Sect. 2, an a priori knowledge graph is costly and tiresome to obtain. Hence, a complete graph is easier to build. However, it also demands a high computational cost and can link incoherent objects, confusing the learning model.

Algorithm 1: Proposed approach

input : set of images \mathcal{S} , a given CNN C
output : learning model \mathcal{M}
auxiliaries: sets of extracted features Z_i , training sets Z' , testing sets Z'' , a vanilla GNN G , a fine-tuned GNN G^Ω

for each $S_j, j = 1, 2, \dots, |\mathcal{S}|$ **do**
 $\mathcal{B} \leftarrow \text{boundingBoxes}(S_k)$;
 for each $\mathcal{B}_i, i = 1, 2, \dots, |\mathcal{B}|$ **do**
 $Z \leftarrow Z \cup \text{deepFeatures}(\mathcal{B}_i, C)$;

Split Z ;
 $Z' \leftarrow$ random training samples from Z ;
 $Z'' \leftarrow$ random testing samples from $Z \setminus Z'$;
 $G \leftarrow \text{completeGraph}(Z', Z'')$;
 $G^\Omega \leftarrow \text{fineTunedGraph}(G.\text{graph})$;

4 Experiments

This section discusses the scenarios of the experiments, describes the image dataset used in the experiments, and presents the discussion about the obtained results comparing our proposed approach against complete and random graphs, respectively.

Algorithm 2: Tuning mechanism

input : an undirected graph \mathcal{G}
output : fine-tuned undirected graph \mathcal{G}^Ω
auxiliaries: \mathcal{V} : set of vertices from the input graph, \mathcal{A} : list of edges incident to vertex v , τ : threshold to suppress irrelevant edges
for each $v \in \mathcal{V}$ **do**
 $A \leftarrow \text{getIncidentEdges}(v)$
 $\tau \leftarrow \text{calculateMeanDistance}(A)$
 foreach $a \in A$ **do**
 if $\text{edgeRelevance}(a) > \tau$ **then**
 $\text{removeEdge}(a)$

4.1 Scenarios

To obtain the best hyperparameters to the GCN we executed a grid-search strategy [1]. To do so, we defined the number of hidden layers (16, 64, 256), epochs (2000), learning rate (0.001, 0.005, 0.01, 0.05), and dropout (0.3, 0.5, 0.8, 0.9). These possibilities of hyperparameters resulted in 384 experiments.

We combined the GNN with different CNN architectures. Then, we used Efficient-NetB7 [20], InceptionV3 [19], ResNet50 [9] and VGG19 [17]. Finally, to corroborate the effectiveness of our approach we compared it against a fully connected and random graphs. As optimizer we used ADAM [11].

To perform the experiments, we used a computer with Intel Core i7 from the 6th generation with 8 cores, 16 threads, and 3.40 GHz; 32 GB of RAM and an Nvidia GeForce RTX 2080Ti GPU, with 4352 CUDA cores. The experiments were executed in GPU mode.

4.2 Dataset Description

For the experiments, the MIT67 [15] dataset was chosen because it provides the requirements for the development of this work, as the: images, bounding boxes of each image, and global classes (image classes).

The MIT67 is a dataset to solve indoor scene recognition, including 67 different classes. However, because some classes have few examples, annotation errors, missing data, or images without bounding boxes, data cleaning was required, which resulted in the exclusion of the following classes: *auditorium*, *bowling*, *elevator*, *jewelry shop*, *locker room*, *hospital room*, *restaurant kitchen*, *subway*, *laboratory wet*, *movie theater*, *museum*, *nursery*, *operating room*, *waiting room*. Thus, obtaining 53 classes, 2607 images, and 50.8 68 bounding boxes, that were divided into the training (80% of the data) and test (20%) sets in a random and stratified way.

Table 1 shows the distribution of images/objects by class and bounding boxes by image. It is possible to note that the dataset is unbalanced, having a high number of samples for some classes, such as: “kitchen” (308 images, 7511 bounding boxes), “bed-room” (350, 5112); and low numbers for others like “cloister” (16, 159) and “winecellar” (16, 222).

Table 1. MIT67 dataset distribution.

Analysis	Mean	Std	Min	Median	Max
Images per class	49.2	68.1	16	19	350
Objects per class	959.8	1311.7	159	475	7511
Objects per image	18.8	12.7	1	17	95

4.3 Results

To perform the experiments, we trained GCNs aggregated with the four CNNs architectures cited in Sect. 4.1. Tables 2 and 3 show the obtained results regarding the fully connected graph against our approach, respectively. We calculated metrics considering the efficacy, such as accuracy, precision, recall, and F1, to analyze the results. We also show the total number of edges generated and the dimensionality of the feature vectors used to represent the nodes.

Table 2. Results for top 1 performance for fully connected graph on MIT67 dataset.

Architecture	Fully connected graph on MIT67 dataset					
	Accuracy	Precision	Recall	F1	Edges (10^5)	Dimension
EfficientNetB7	69.90	58.68	58.30	56.33	13.95	2560
InceptionV3	63.47	55.67	56.10	52.61	13.95	2048
ResNet50	64.89	56.57	52.69	52.22	13.95	2048
VGG19	57.81	47.90	44.80	42.98	13.95	512

Analyzing Tables 2 and 3 we can see that our approach statistically ties with the vanilla graph (fully connected graph). Moreover, our approach decreased to a great extent the number of edges of the graph.

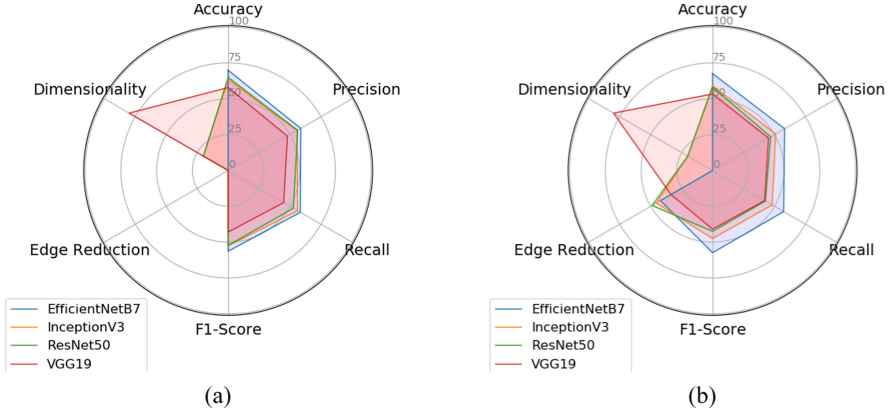
For instance, the fully connected graph and our approach presented 13.95×10^5 and 7.13×10^5 , respectively, regarding the ResNet50. Thus, our approach reduced the memory footprint up to 96%. We observed this same behavior when analyzing the other CNNs aggregated with our GNN. According to the results, our approach with EfficientNetB7, InceptionV3, and VGG19 achieved reductions of up to 73%, 83%, and 51%, respectively.

To better visualize all the considered metrics, in Figs. 3 (a) and (b), we generated the radar plots with the obtained results regarding the fully connected graph and our approach, respectively. It is clear to note that our approach accomplished a considerable edge reduction while maintaining efficacy.

We also performed experiments considering random connections. To create the random edges and obtain a fair comparison, we used the same number of edges obtained by our approach. It is important because using a higher or lower number of random edges, when compared with ours, could lead to a false degeneration of the graph or a false improvement (higher computational cost). Thus, our approach can also answer which is the best number of edges to reach a suitable trade-off between efficacy and efficiency.

Table 3. Results for top 1 performance considering our approach on MIT67 dataset.

Architecture	Our approach on MIT67 dataset					
	Accuracy	Precision	Recall	F1	Edges (10^5)	Dimension
EfficientNetB7	67.58	58.18	57.37	57.37	8.08	2560
InceptionV3	59.10	50.94	48.07	47.49	7.62	2048
ResNet50	58.21	46.93	42.67	42.50	7.13	2048
VGG19	53.33	45.17	42.06	40.91	9.26	512

**Fig. 3.** Radar plot using different CNN architectures; (a) Fully connected graph; (b) Pruned graph.

The experiment regarding the random edges achieved an accuracy of up to 60% with InceptionV3. The same behavior, where our approach presented the best results, was observed regarding the other CNNs. This testifies that our approach effectively defines the edges' relevance, capturing the semantic relationship between the objects of an image to define its global context.

Thus, considering the obtained results, we can argue that our proposed approach was capable of automatically tuning the GNN graph structure aggregated with a given CNN, defining well-suited connections according to a given context, improving the entire process.

5 Conclusions

In this paper, we proposed an approach capable of automatically fine-tuning a given GNN. To do so, it defines the well-suited connections according to a given image context, improving the effectiveness of the entire process. Our approach was based on the similarity between the graph nodes (i.e., images).

Despite its simplicity, it reached considerable results. It not only successfully reduced the memory footprint when using GNNs aggregated with CNNs, but also maintained good accuracies when compared with random and complete graphs. This testifies

that our approach achieved better representativeness of the semantic between an image and its objects to define the global classification context. Our results showed that the fine-tuned GNN reached up to 96% regarding the memory footprint while maintaining the accuracy.

For future works, we intend to explore other image datasets. We also aim to extend our approach regarding different distances and detection mechanisms for edge relevance.

References

1. Andonie, R.: Hyperparameter optimization in learning systems. *J. Membr. Comput.* 1–13 (2019). <https://doi.org/10.1007/s41965-019-00023-0>
2. Bronstein, M.M., Bruna, J., Lecun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. *IEEE Sig. Process. Mag.* **34**(4), 18–42 (2017)
3. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and deep locally connected networks on graphs. In: *Proceedings of the International Conference on Learning Representations*, pp. 1–14 (2014)
4. Chen, X., Gupta, A.: Spatial memory for context reasoning in object detection. *CoRR abs/1704.04224* (2017)
5. Chen, X., Li, L., Fei-Fei, L., Gupta, A.: Iterative visual reasoning beyond convolutions. *CoRR abs/1803.11189* (2018)
6. Chung, H., Huang, H., Chen, H.: Predicting future participants of information propagation trees. In: *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 321–325 (2019)
7. Gallicchio, C., Micheli, A.: Graph echo state networks. In: *Proceedings of the International Joint Conference on Neural Networks*, pp. 1–8 (2010)
8. Gori, M., Monfardini, G., Scarselli, F.: A new model for learning in graph domains. In: *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 729–734 (2005)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. *CoRR abs/1512.03385* (2015)
10. Hong, L., Zou, L., Lian, X., Yu, P.S.: Subgraph matching with set similarity in a large graph database. *IEEE Trans. Knowl. Data Eng.* **27**(9), 2507–2521 (2015)
11. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014)
12. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: *Proceedings of the International Conference on Learning Representations*, pp. 1–14 (2019)
13. Li, Q., Han, Z., Wu, X.: Deeper insights into graph convolutional networks for semi-supervised learning. *CoRR abs/1801.07606* (2018)
14. Marino, K., Salakhutdinov, R., Gupta, A.: The more you know: Using knowledge graphs for image classification. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 20–28 (2017)
15. Quattoni, A., Torralba, A.: Recognizing indoor scenes. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 413–420 (2009)
16. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. *IEEE Trans. Neural Netw.* **20**(1), 61–80 (2009)
17. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556* (2014)

18. Sperduti, A., Starita, A.: Supervised neural networks for the classification of structures. *IEEE Trans. Neural Netw.* **8**(3), 714–735 (1997)
19. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. *CoRR abs/1512.00567* (2015)
20. Tan, M., Le, Q.V.: Efficientnet: rethinking model scaling for convolutional neural networks. *CoRR abs/1905.11946* (2019)
21. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. *CoRR abs/1901.00596* (2019)
22. Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Sun, M.: Graph neural networks: a review of methods and applications. *CoRR abs/1812.08434* (2018)