Graph Neural Networks for Scene Segmentation

Nicholas Ramirez's Computer Vision submission

Abstract

Scene segmentation is a developing field of research that affects a wide range of domains such as medical imaging, self-driving cars, and robotics. This paper presents the application of a graph neural network in the computer vision domain. Specifically, the paper proposes a conversion method to utilize a message-passing Graph Neural Network in order to complete the task of scene segmentation. This new method potentially enables the future use of non-standardized image shapes in scene segmentation. My empirical results show that the direct application of GNNs to scene segmentation leads to poor performance. However, this paper outlines potential reasons why and further areas to explore in order to apply GNNs to the scene segmentation task.

1. Introduction/Motivation

Many different fields such as medical imaging, self-driving cars, and facial recognition utilize some form of scene segmentation in order to complete complex tasks accurately. Scene segmentation, also known as pixel-wise dense labeling, is the task of splitting an image into various object components by assigning a label to each pixel. This is one step further from the typically object detection where you detect and separate general objects within an image. By labeling each pixel, the task produces a much finer and potentially more accurate depiction of objects in the image which can be utilized for much more complex tasks such as self-driving cars. There are two types of segmentation in today's literature: scene (semantic) and instance segmentation. Scene segmentation is labeling general object categories to each pixel whereas instance segmentation is labeling different instances of an object within an image. This paper will focus on scene segmentation.

Computer vision has seen rapid recent developments within scene segmentation, but most of the literature utilizes common models derived from other various computer vision tasks such as convolution neural networks or transformers. This paper implemented and tested a Graph Neural Network (GNN) [3] in order to accomplish scene segmentation. Graph Neural Networks are machine learning applications centered on passing information between nodes in order to create a node embedding where more similar nodes are represented by more similar embeddings. These models have seen significant performance in node classification problems, especially in the biomedical domain. Due to the recent successes of graph neural networks, this paper explores the potential uses within scene segmentation as images can naturally be converted into a graph structure. The ultimate goal of this paper is to explore the overall utility of graph neural networks within scene segmentation. This can enable the future use of non-standardized forms of images such as hand sketches where the implicit need for a rectangle shaped image isn't necessary.

2. Related Works

There has been a lot of literature on scene segmentation, specifically surrounding different models such as convolutional networks and transformers. Most of the literature is centered on deep learning, such as Fully Convolutional Networks [6], PointNet [7], and SegNet [2]. All these related works follow the same deep learning approaches that has seen success for years such as convolutional networks, transformers, and encoder-decoder setups.

For graph learning domain, the recent success with graph neural networks has led to a large amount of literature on different models. These range from Graph Convolutional Networks [3], Graph Attention Networks [9], and Graph Partition Neural Networks [4]. The main tasks for these models that is relative to this paper is node classification which is predicting the label for a given node [10].

In terms of using Graph Neural network for scene segmentation, a recent paper [1] utilizes a Vision Transformer to create a deep feature that the GNN can use to ultimately cluster the nodes. This paper is unsupervised learning, but in our datasets, we have labels. As a result, my paper will utilize the GNN for a larger role than simply clustering with N-cuts method.

3. Method

The goal is to extend the GNN into the computer vision scene segmentation task. In order to investigate its full capability within this domain, I first need to validate the GNN to ensure the following experiment is accurate. To do so, I ran the Graph Neural Network on the Cora dataset [8] which is a graph dataset that consists of 2708 scientific publications and seven total potential classes. These experiments will compare my model to some baseline to ensure expected results to validate the entire setup is working correctly. The baseline I used is a simple Multilayer Perceptron (MLP) model. We expect this baseline should perform worse for node classification since no additional graph information such as the neighborhood information.

For my main experiment, I directly apply this GNN model to the scene segmentation task. In order to accomplish this, I first had to convert images from the ADE20k dataset [11] into a graph-like structure. Specifically, for my data, I am using the Scene Parsing Benchmark version which contains 20,210 training images, 2000 validation, and 150 object categories. This dataset provides a nice range of image shapes and non-uniform distribution of objects within the images themselves. In order to utilize a GNN on these images, I had to preprocess the images by converting them into graphs. To convert an image into a graph, our nodes for the graph will simply be pixel locations with the node features being the RGB values. Then edges are added to our graph if the two nodes are neighboring pixels in the image. These edges are undirected since the relationship is symmetric. This is shown in Figure 1.

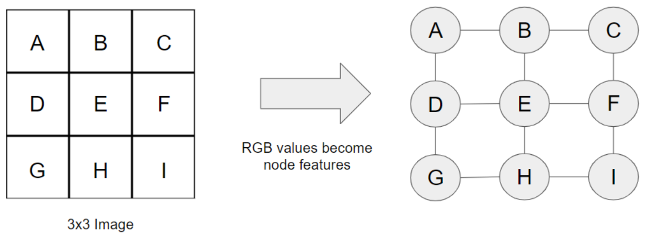

Figure 1. A small example of image to graph conversion.

For the framework of my model, I used a conventional Graph Neural Network that utilizes message-passing to create node representations. The internal goal of a GNN is to create internal node representations for each node that can be used to ultimately classify that node. These node representations are created and updated using an iterative message-passing scheme which contains two parts: an aggregation and update step. The aggregation step is simply the sum of all neighborhood node representations, and the update step is updating the current node's representation with this new aggregated information. These iterative steps are completed a number of times depending on the total layers of the model. For scene segmentation, message-passing theoretically provides a potentially compatible framework as a given node (pixel) will utilize information from its neighbors. Since the task inherently has objects in an image being a clump of pixels, this information is necessary and useful. The overall message-passing scheme is shown in Figure 2.

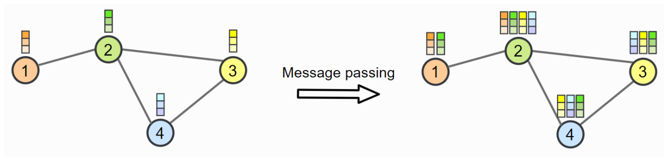

Figure 2. A small example of message-passing within a Graph Neural Network as taken from [5].

With the model defined, we apply the GNN to the scene segmentation task by utilizing the already common task of node classification. Since the nodes of our graphs are pixels, I directly translate the the node classification task into the pixel labeling task. Here, the nodes are being classified into the 150 potential objects, and each classification is our attempted label to that pixel. Thus, the performance metric I utilize is accuracy from the node classification task.

4. Results

The first experiment mentioned in methods is the validation of my GNN model. This validation is comparing the accuracy for semi-supervised node classification between my GNN model and a MLP baseline. Table 1 shows these results.

Table 1. Validation of GNN on Cora Dataset.

Model	Train Acc	Val Acc	Test Acc
MLP (baseline)	96.43	52.60	60.60
GNN	99.29	78.00	81.10

As expected, the GNN outperforms the MLP and achieves an expected level of performance on a graph dataset. Thus, the model is properly implemented. Since this experiment is validating my setup, there isn't much insight or discussion.

For the main experiment, I ran into very difficult obstacles involving graph size for the neural network. Graph Neural Networks are designed for sparse graphs that usually contain less than 100,000 nodes. A 500x500 image (which is considered a small image) has 250,000 total pixels (nodes) and almost 1,000,000 edges. As a result, I ran into many memory issues when attempting to complete scene segmentation task. Due to the short time constraint of the project, I created a small dataset from ADE20k that consists of only images with size less than (300, 300). There was a total of 4,122 images that fit this criteria. After running a hyperparameter search, the corresponding best parameters were a hidden dimension size of 16, 2 total layers, drop out rate of 0.1, Stochastic Gradient Descent optimizer, ReLU activation function, and learning rate of 0.1. The best training, validation, and test accuracy are shown below in Figure 2. In addition, Figure 3 makes it clear that the performance of the model is poor for a given image as there is no clear labeling between objects.

Table 2. Results from GNN on scene segmentation.

Model	Train Acc	Val Acc	Test Acc
GNN	26.88	20.35	19.69

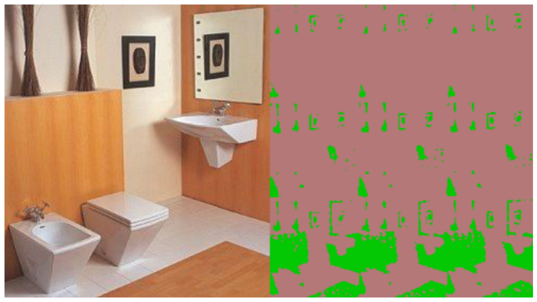

Figure 3. GNN's image results for the scene segmentation on a randomly chosen image from the validation set (image 116).

5. Discussion

The results of the main experiment can be broken down into two main categories: the performance of the model on images and image size obstacle. For the performance of the model within the image domain, my work empirically shows that the direct application of GNNs to the scene segmentation task leads to poor results relative to state of the art performance. There are two main reasons I think could have contributed to the poor results. The first reason is my choice of filtering for small images within the dataset to overcome computational limitations. By doing this preprocessing, our total dataset size significantly decreased from over 20,000 training images to just over 4,000. This could have less object categories being seen in training and then tested with the test set. The last possible reason is the use of message-passing for scene segmentation might not be the best framework. One of the most significant benefits from message-passing is gaining information about the total graph structure as messages are only passed if an edge exists. With our image converted graphs, the graph structure is completely balanced between all graphs (mostly 4 edges between all neighboring pixels). The overall graph structure is already known and follows this pattern, so message-passing might not gain much information. In addition to this issue, the message-passing framework might only work well with smaller graphs where a couple hops (layers) is sufficient information. In the case of images, 2 hops translates to only 2 pixels away which is relatively very small compared to the whole image. As a result, it is very susceptible to noise that naturally can occur in images which can lead to significant overall performance changes. In order to increase these hops, that requires significant computation by adding more layers which is not possible since computation is already a limitation of this approach as seen with image sizes. In addition to poor performance, my work reveals that the inherent obstacle of image size and the overall computation limitations of GNN requires more work and exploration. The overall image size in this ADE20k dataset is relatively small as the standard HD image size for pictures is 1280 x 720 pixels. In general, the dataset was around had around 600x600 dimension for images. As a result, this computation limit needs to be addressed in order for the GNN to be applied in any meaningful way. Ultimately, these results lay a foundation for applying GNNs to this task as no other work in the previous literature uses GNNs as the main component for image scene segmentation.

6. Conclusion

Given the importance of scene segmentation for computer vision tasks, this paper explored the potential usage of Graph Neural Networks within this new domain. This was accomplished by reducing the scene segmentation task to a well known node classification problem as described in the method. Ultimately, the application of a message-passing GNN led to overall poor results for the scene segmentation task. However, there is still many potential ways GNNs can be utilized in computer vision tasks. As mentioned within the discussion, there are still many open problems. The first and most important is overcoming the computational limit of GNN with large image sizes. For future work, I would suggest using a sampling method to reduce the image size or using an object detection model to get a bounded box over objects, and then using a fine-tuned GNN to label the pixels within these much smaller bounded box images. In addition to the image size problem, the second open problem is to determine the best GNN framework to utilize for the scene segmentation problem. The main experiment revealed that message-passing may not be the best solution, but there are many different approaches to GNNs that might lead to better performance. For future work on the scene segmentation task, I would suggest utilizing a non message-passing model so the neighborhood information used by the model is greater than its immediate neighbors.

References

[1] Amit Aflalo, Shai Bagon, Tamar Kashti, and Yonina Eldar. Deepcut: Unsupervised segmentation using graph neural networks clustering, 2022. 1

[2] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *CoRR*, abs/1505.07293, 2015. 1

[3] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017. 1

[4] Renjie Liao, Marc Brockschmidt, Daniel Tarlow, Alexander L. Gaunt, Raquel Urtasun, and Richard Zemel. Graph partition neural networks for semi-supervised classification, 2018. 1

[5] Phillip Lippe. UvA Deep Learning Tutorials. https://uvadlc-notebooks.readthedocs.io/en/latest/, 2022. 2

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. 1

[7] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016. 1

[8] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008. 2

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

[10] 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. 1

[11] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5122–5130, 2017. 2