Improving Image Segmentation with Edge Detection and Filters

Jing Lin
Massachusetts Institute of Technology
77 Massachusetts Ave, Cambridge,
MA 02139
jingglin@mit.edu

Shang-Yun (Maggie) Wu
Massachusetts Institute of Technology
77 Massachusetts Ave, Cambridge,
MA 02139
maggiewu@mit.edu

Abstract

Image or scene segmentation is a rapidly-moving field in computer vision. It focuses on separating objects along contours in a given image. Traditional methods include gray-scale image thresholding and clustering. Recently, there have been breakthroughs using Fully Convolutional Networks (FCN) and Regional Convolution Neural Networks (R-CNN). In this paper, we follow the architecture proposed by MIT CSAIL Vision Group, which utilizes the ADE20K Dataset as training and validation data for their encoder-decoder structure [6]. On top of the pretrained model, we modify inputs based on edge detection techniques and common filters to enhance the boundaries of individual objects. We further compare the results and explain how some of these techniques can help improve scene parsing.

1. Introduction

To this day, many autonomous vehicles and robots rely heavily on computer vision for navigation. They decide their paths and operate on things given vision information. In order to do so, these machines need much more than just the ability to take photos and visualize general image. For example, a robot needs to recognize the handle of a cup as part of the cup in order to lift it, and an autonomous car needs to segment the sidewalk from the road in order to stay on track. These come down to the ability to segment objects and scenery. Therefore, scene segmentation remains a hot topic due to its wide applications.

Different from standard Convolution Neural Network (CNN) where one simply classifies images based on its identity (e.g. dog, cat, plane), scene segmentation requires detailed object annotation. Many previously used datasets do not include enough information of common objects one would see in real life or are too specific for particular scenery (e.g. airport, parking lot, playground). These make image segmentation difficult for the purpose of autonomous machines in daily lives.

The state-of-the-art segmentation network identifies the core pixels of most objects correctly but fails to distinctly recognize the boundaries of the objects detected [6]. Thus, our work aims to enhance edge recognition in segmented objects by merging edge detection and filter techniques with developed segmentation module.

The paper is organized as follows. First, we discuss our segmentation techniques, which harness the segmentation module provided by MIT CSAIL Vision Group along with edge detection and filters. Then, we present our experiment results, where a set of images from the ADE20K validation set are selected for comparison. The table comparing the performance of the original model and our technique vs. the ground truth is shown. Moreover, specific part where our techniques perform badly and well is analyzed. Finally, we give thoughts on future work to be done that can potentially increase scene parsing accuracy.

1.1. Related Work

Traditionally, many work and literature discuss about image segmentation using thresholding and clustering methods. Since they are not explicitly related to our work, details of these methods are neglected. We, instead, review the ADE20K dataset that is being used and briefly discuss the three other increasingly popular methods, FCN, R-CNN and Boundary Detection Image Segmentation (BDIS).

ADE20K Dataset includes a total of 20,210 images in the training set, 2000 images in the validation set, and 3000 images in the test set. Most importantly, these images are exhaustively annotated by expert annotators and come from a range of 900 scene categories. Thus, the completeness and consistency make it a good dataset to study.

A **FCN** is a type of CNN without any fully-connected layer throughout the entire architecture. Given this design, the architecture focuses on learning local filters that help with global decision making. Since image segmentation concerns with local objects, using FCN enables learning features that are useful for determining instances in given images [4].

A **R-CNN** classifies instances via bounding boxes with labels. It does so by proposing possible bounding boxes via *Selective Search* and checking if the enclosed areas entail any instances. To do so, corresponding segments are passed into a classification network; the classification network can then identify the object [2].

BDIS is a newly developed technique that applies idea similar to that of our work. There are two major parts to the technique, the image segmentation neural network and the boundary detection neural network. These two merge to be a deep convolution neural network (DCNN) where the boundary detection network integrates with the image segmentation network; the entire DCNN is trained altogether for greater precision. Because this network considers both boundary and image segmentation, it is able to overcome the limitation associated with unclear object boundaries [3].

2. Segmentation Techniques

The segmentation module used in our work is one of the pretrained models *(ResNet50dilated Encoder + PPM_deepsup Decoder)* presented by MIT CSAIL Vision Group. Images are processed via the following techniques to enhance the instance boundaries and are then passed into the model for image segmentation.

2.1. Edge Detection

To effectively determine the performance of variants of edge detection technique, we process images in various ways upon edge detection. Specifically, we use the *Canny Edge Detection* method. It is known to perform better than most standard edge detection methods as it is less sensitive to noise; that is, fewer false edges are detected [1]. All of the techniques introduced below make use of this method for image processing.

The **Thicken** (**T**) method takes edge detection results from the original image and thickens each of these edges. For any edge pixel, the algorithm assigns the 9 pixels around, including itself, to be the same color as the edge pixel. In other words, it extends the width of all edges and makes them more substantial. In the case of conflicting (i.e. overlapping) assignments, the color is randomly assigned to one of the edge pixels.

The **Darken-Thicken** (**DT**) method is exactly the same as the Thicken method, except that the algorithm assigns the surrounding pixels to have color that is 20 units lower on each of the rgb scale, therefore making the edges darker and more distinctive.

The **Pure-Darken** (**PD**) method focuses on the exact edge pixels and darkens the pixels by 20 units on each of the rgb scale. In which case, the edge pixels stay contained as they are identified by the edge detection mechanism.

2.2. Filter

The **Kernel** (**K**) method uses a 3×3 sharpen filter to enhance the definition of an image. The sharpen filter takes the following form.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

The sharpen filter is applied through every pixel in the image where border type is constant along image borders.

The **Contrast** (**C**) method sets a threshold value such that pixel values change with respect to its difference to the threshold value. In which case, the high pixel values will become even higher and low pixel values will become even lower, thus increasing the brightness contrast. Threshold values of 50 (**C-50**) and 100 (**C-100**) are used.

2.3. Combination of Techniques

The **Darken-Kernel** (**DK**) method darkens the image via the Pure-Darken method as described earlier then applies the Kernel method to the image.

The **Kernel-Darken** (**KD**) method is a flip of Darken-Kernel method in which the algorithm applies the Kernel method then darkens the edges via the Pure-Darken method.

3. Experimental Results

Several images from different categories of the ADE20K validation set are chosen for comparison. To illustrate the effects of our techniques, the results for two selected images (*Bread* and *Pool*) are presented in Table 1, where the row of **Baseline** (**B**) images is the original segmentation outcome without modification.

For error measurement, every pixel value of segmented image output is compared to the ground truth labelled by humans. The specific error metric used is Mean Squared Error (*MSE*) in which all misclassified pixels contribute to the error with respect to their distance from the true value.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \left(y - \hat{y}\right)^2 \tag{1}$$

where y is the ground truth pixel value and $\hat{y}$ is the output pixel value.

Using MSE penalizes less for almost correct labellings and more for labellings that are far off. This loss function assesses not only the correctness but also the quality of the output in relation to the ground truth.

Structural Similarity (*SSIM*), which is widely used for images comparison, is also used as another metric in the result analysis to account for global comparison between two images instead of only pixel-wise local information as done via MSE. [5]

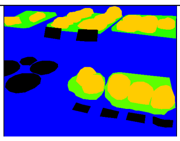
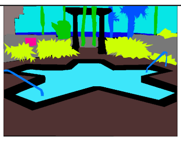
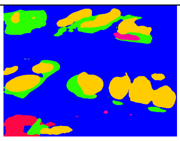
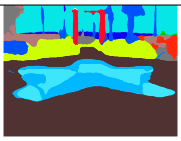
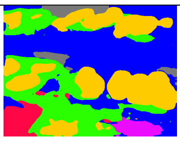
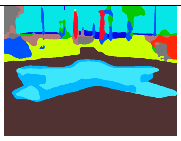
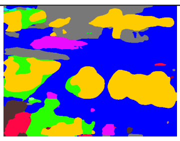
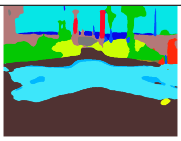
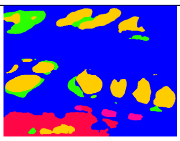
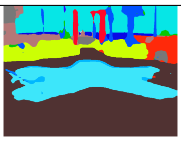
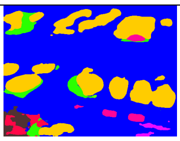
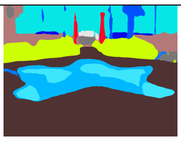
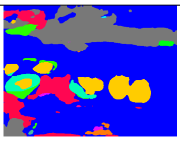
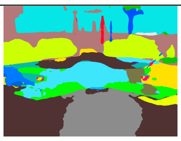
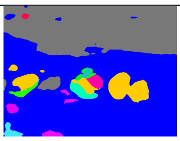
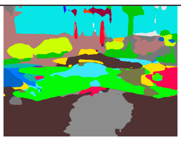
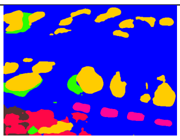
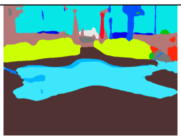
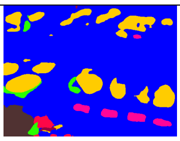
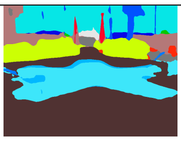
Original		
Ground Truth		
Baseline		
Thicken		
Darken-Thicken		
Pure-Darken		
Kernel		
Contrast-50		
Contrast-100		
Darken-Kernel		
Kernel-Darken		

Table 1. Selected image segmentation results (*Bread* and *Pool*), shown in the order in which the methods are discussed.

4. Discussion

Given the experimental results, MSE and SSIM values measured across the selected images are shown in Table 2.

Metrics	MSE	SSIM
Baseline	9.60e-4	0.74
Thicken	1.07e-3	0.72
Darken-Thicken	1.11e-3	0.69
Pure-Darken	9.89e-4	0.74
Kernel	1.15e-3	0.72
Contrast-50	1.27e-3	0.66
Contrast-100	1.36e-3	0.62
Darken-Kernel	1.17e-3	0.69
Kernel-Darken	1.21e-3	0.69

Table 2. MSE and SSIM comparison.

The corresponding MSE and SSIM are shown as bar graphs, in Figure 1 and Figure 2, respectively, to illustrate performance variances for each of the technique. The MSE values suggest that the baseline model still performs the best overall. However, the performances of Thicken, Pure-Darken and Kernel techniques come close to the baseline model. As SSIM examines structural similarity between images, we can see similar effect to that of MSE on SSIM across various techniques where Thicken, Pure-Darken and Kernel techniques work relatively well.

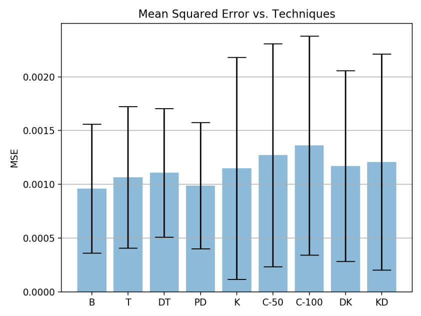


Figure 1. Mean Squared Error bar chart

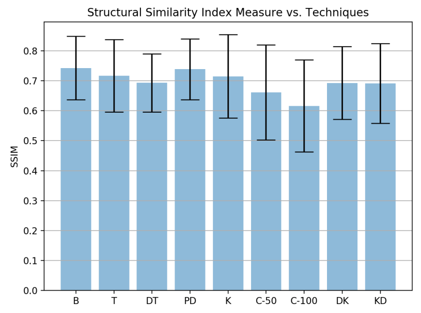


Figure 2. Structural Similarity bar chart

A reasonable explanation for the high MSE and low SSIM value would be the lack of training with respect to each individual method due to time limitation. Our images are modified and fed into the model that is trained using original images. Essentially, we are evaluating the model performance using images that are different from what the model is trained for. Therefore, it is likely that training a new model for each technique would yield higher accuracy.

Nevertheless, the results obtained suggest benefits in some of the technique presented. Specifically, in Table 1, one can recognize that there are parts of the images where the segmentation works better in proposed techniques than in the baseline model. For instance, the segmentation on the *Bread* image can be seen more clearly via Kernel technique (i.e. the Kernel technique detects the bread on the top right corner of the image which baseline fails to do). Meanwhile, the segmentation on the *Pool* image works well with Pure-Darken technique (i.e. Pure-Darken outlines the boundary of the pool such that the pool is seen as one single object).

Above analysis suggests that, despite the baseline model working best, using the techniques discussed can potentially still result in better identification for particular segments of an image. This means that we can expect the techniques to outperform the baseline model by analyzing how they fit well, whether it is in brightness, shape or other structural features of an image. More specifically, if one technique improve some parts of image segmentation, we can always identify and pre-process those segments of an image before feeding it into the segmentation network, thus increasing the overall accuracy.

5. Conclusion

In this paper, we present the edge detection and filter techniques that can be utilized in addition to the pretrained segmentation module. The techniques in a nutshell enhance segmentation by placing greater emphasis on edges in the hope to reduce uncertainty with pixels near boundaries of any object. Specifically, all of the edge detection or filter methods are applied to the original image before feeding into the segmentation module.

The experimental results demonstrate places in which our techniques can be useful. Thus, with suitable image pre-processing and proper training, we can anticipate better image segmentation results.

6. Future Work

A potential future work is analyzing the effects of each technique on performance for parts of the image. The result of this analysis would help with future image pre-processing. It also helps to retrain the model based on the pre-processed images such that the network accounts for the way images are pre-processed.

Acknowledgment

This work was supported by staff of the MIT course 6.819 *Advances in Computer Vision*. We thank them for providing insight and guidance that assisted with this project.

Contribution

Both authors contributed equally to this work, including the code, image analysis and report. Both authors reviewed the manuscript.

My specific contribution to this project includes creating most edge detection and filter techniques. Additionally, I provide Figure 1 and 2 and Table 2 for error analysis.

References

[1] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, June 1986.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *ArXiv e-prints*, page arXiv:1311.2524, 2013.

[3] D. Marmanis, K. Schindler, J. D. Wegner, S. Galliani, M. Datcu, and U. Stilla. Classification with an edge: Improving semantic image segmentation with boundary detection. *ArXiv e-prints*, page arXiv:1612.01337, 2016.

[4] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, 2017.

[5] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 13(4):600–612, 2004.

[6] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ade20k dataset. *ArXiv e-prints*, page arXiv:1608.0544, 2016.