Performing Semantic Segmentation on an Extremely Small Dataset

Hao Du, Rodney LaLonde, Ryan van Mechelen, Skylar Zhang
Department of Computer Science,
St. Olaf College, Northfield, MN

Abstract

Semantic segmentation is a difficult task, and even more difficult with limited data. In this paper, we employ two different approaches to semantically segment 30 images of dimension 1188×792 pixels. Our first method uses a convolutional neural network to train a classifier on subimages of 11×11 pixels. The second method involves training on a classifier on 20×20 pixel subimages, but using a different convolutional network of one convolutional layer with batch normalization and ReLU, a new deconvolutional layer and one loss layer. The first approach produces a somewhat coarser segmentation but fairly high accuracy, 68 percent. The second method, employing the new architecture, results in much crisper lines for a finer segmentation, but a lower accuracy.

1 Introduction

1.1 Semantic Segmentation

Humans excel at breaking down a scene into its separate objects. However, computers have a much harder time doing so. Deep learning is a powerful tool improving computer ability to perform semantic segmentation, which is classifying each pixel of an image. For optimal results, this involves the challenge of combining both the global context and local characteristics of an image together. Since convolutional neural networks work well in capturing the global spacial structure of an image, they are useful for the task of semantic segmentation. Generally, large datasets are required to train high-accuracy classifiers. In this paper, we seek to demonstrate that reasonable accuracy and semantic segmentation can be achieved on an extremely small and non-ideal dataset. We used a modified Caffe framework to construct a simple architecture for semantic segmentation with the building block of one convolutional layer and a loss layer to accomplish this task.

1.2 Related Work

There have been many approaches to tackle semantic segmentation, with new strategies constantly outdoing old ones. Due to the newness of the field, there are many areas yet to explore. One approach by Jon Long *et al.* uses fully convolutional neural networks (FCNs) with a deconvolutional layer and skip architecture to adapt a CNN that was successful for object recognition to semantic segmentation[2]. This paper introduced an architecture for semantic segmentation combining both a fully convolutional network and a deconvolutional network. The architecture includes batch normalization as part of a convolutional layer as well as a deconvolutional layer. Another strategy has been to use conditional random fields (CRFs) to determine local segmentation labels to combine with global classification labels that indicate which classes are present in the image before segmentation[4]. In this way, the local contextual information is considered with the global contextual information. Furthermore, another approach is to utilize Regions with Convolutional Neural Network features (R-CNN) with boundary box proposals.

1.2.1 Batch Normalization

Batch normalization is known as a technique to solve the internal-covariate-shift problem [1]. We put a batch normalization at our convolution layer to normalize the the every layer to the standard Gaussian distribution, as done when one is training very deep neural network[3]. The output image with no batch normalization in the training stage is chaotic, as shown in Figure 8.

2 Method

2.1 Training Data

We have a minimal training dataset of only 30 images with the ground truth of our training data created manually. In order to augment the amount of training images, we split each image into sets of 11×11 pixel subimages for the first method and 20×20 subimages for the second. As such, we generated more than four hundred thousand and two hundred thousand training images respectively. Splitting the image was done by copying the 20×20 pixels from the original image, then sliding over with a horizontal stride of 1 and creating another 20×20 image. At the end of a row, we move down with a vertical stride of 1, until the entire image had been traversed. We also used this strategy to create a training set of 11×11, using instead 8 as the horizontal and vertical stride. In total, we trained on 5 classes for the first method and 35 classes for the second method.

2.2 Network Structure

Our first method used a deep convolutional neural network consisting of nine hidden layers. In the second method, we constructed an architecture with a single convolutional layer which is referenced in the Appendix. For the second method, we compared the final segmentation results based on the changes in resize-dimension and adding batch normalization. We first changed the resize-dimension in our data layer: 20×20, 100×100, 500×500. Secondly, we included batch normalization or ReLU in each convolutional layer and compared the results between whether batch normalization or ReLU are performed.

3 Results

3.1 First Method

In the first method, we found that our object recognition classifier achieved an accuracy of 68 percent on the 11×11 pixel subimages and an accuracy of 71 percent on the 30×30 pixel subimages. Interestingly, the additional size of the subimages boosted the accuracy, but not by much. Therefore to achieve a finer segmentation we moved forward only with the 11×11 pixel subimages. These accuracies are quite low to apply ideal semantic segmentation; however, this can be explained by our training data. When we compare our training images to those of the CIFAR-10 dataset (that have similar dimensions), we notice that our training images contain less edges, features, and texture. In fact our subimages contain almost none. Filters used in convolutional layers of a convolutional neural network(CNN) pick out exactly these lines and silhouettes. Since our training images rarely contain much texture that can give clues to the CNN and often contain a solid color, we are dealing with a much more challenging problem. Given that color is largely what might be used to differentiate classes, we plotted the average intensity of each color channel per pixel per class below in Figures 1, 2 and 3. There is considerable overlap between these colors, which demonstrates the extent to how difficult it is to achieve a high accuracy classifier. That said, our semantically segmented images are quite reasonable. The reader can see our segmented images in the Appendix on pages 11 and 12.

3.2 Second Method

In the second method, considering the quality of our result, we found that training on images of 500×500 provided the most fine semantic segmentation with the most classes. While this took considerably more training time, the results differ quite substantially. Below in Figures 5, 6 and 7 are a comparison of training with 20×20, 100×100, 500×500. Though the input data are all 20×20 image sets, providing the same information for all training stages, the resizing may result in a larger matrix output in the first convolutional layer. As the updating requires information from matrix output in each layer, it may reflect more delicate changes in training stage. The resize-dimension has significant impact on the training speed. Training time is an issue. While training on the 20×20 and 100×100 images goes smoothly within a few hours, training on the 500×500 can take more than 20 hours. This is not unreasonable in the field to have long training times, but it is notable in comparison to our other training options. The batch normalization gives a more realistic result. As introduced above, batch normalization keeps the mean and standard deviation consistent throughout the updating stages of each layer. As a result, inclusion of batch normalization give more steady and realistic result. We may compare the results as in figure 6 below for 100×100. The result without batch normalization appears to be extremely colorful, and we interpret the result as not desirable, since the result is chaotic. Moreover, the batch normalization gives much improvement in convergence. As for the effect of ReLU layer, we came to the conclusion that this layer more effectively sketch out the silhouette of different classes. We may see the effect clearly as in figure 5, 6, 7 below.

While the results are not perfect, considering the small data size and our method, they are reasonable. We find that an approach similar to ours seems viable, and can perhaps meet the expectations of some users in a fundamental and easy-to-understand manner. Semantically segmenting a large number of classes is not feasible with this method, but it appears possible with a smaller number of classes.

3.3 Comparison

The first method is a local approach that make the semantic segmentation by examining the pixels of an image, while the second method is a global approach that focuses more on the overall spacial structure of the image. Interestingly, the first achieved a high accuracy and a somewhat course segmentation while the second method is not able to produce a well-defined quantitative accuracy but produced a finer segmentation. We find this counter-intuitive from the current knowledge on performing semantic segmentation and is worth further investigation. Another point of difference is training time. The first method takes very little time to train compared to the second method. Though the first method can produce desirable result as The second method needs long time for training. It costs more then 24 hours to train a 500 times 500 model for 10000 iterations, but the inference for a single images only takes less then 5 seconds. In addition, the second method can produce images with finer details compared with the first one.

References

[1] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[2] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *arXiv preprint arXiv:1411.4038* (2014).

[3] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. "Learning Deconvolution Network for Semantic Segmentation". In: *International Conference on Computer Vision (ICCV).* 2015.

[4] Nils Plath, Marc Toussaint, and Shinichi Nakajima. "Multi-class image segmentation using conditional random fields and global classification". In: *Proceedings of the 26th Annual International Conference on Machine Learning.* ACM. 2009, pp. 817–824.

4 Appendix

4.1 Color Graph

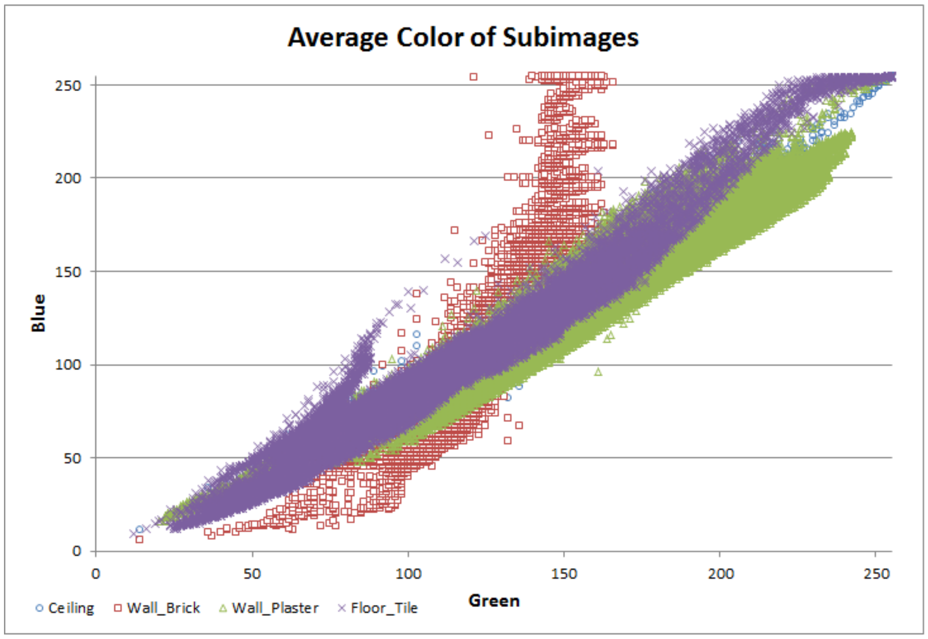


Figure 1: Green Vs Blue Color

4.2 Architecture

Name	Kernel Size	Stride	Pad	Output Size
input	-	-	-	$20 \times 20 \times 3$
conv1_1	3×3	1	1	$500 \times 500 \times 64$
output	1×1	1	1	$500 \times 5020 \times 35$

4.3 Inference Results

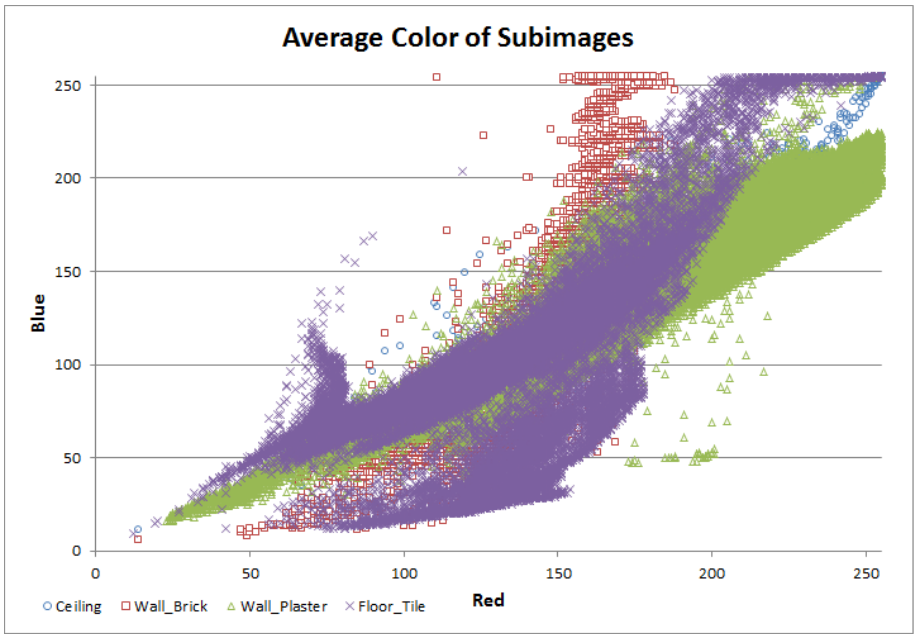


Figure 2: Red Vs Blue Color

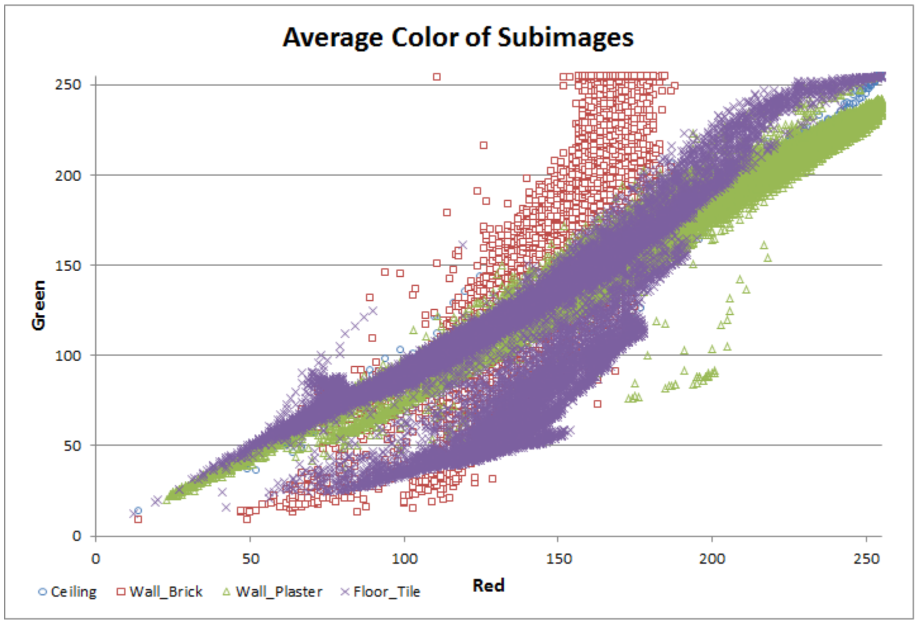


Figure 3: Red Vs Green Color

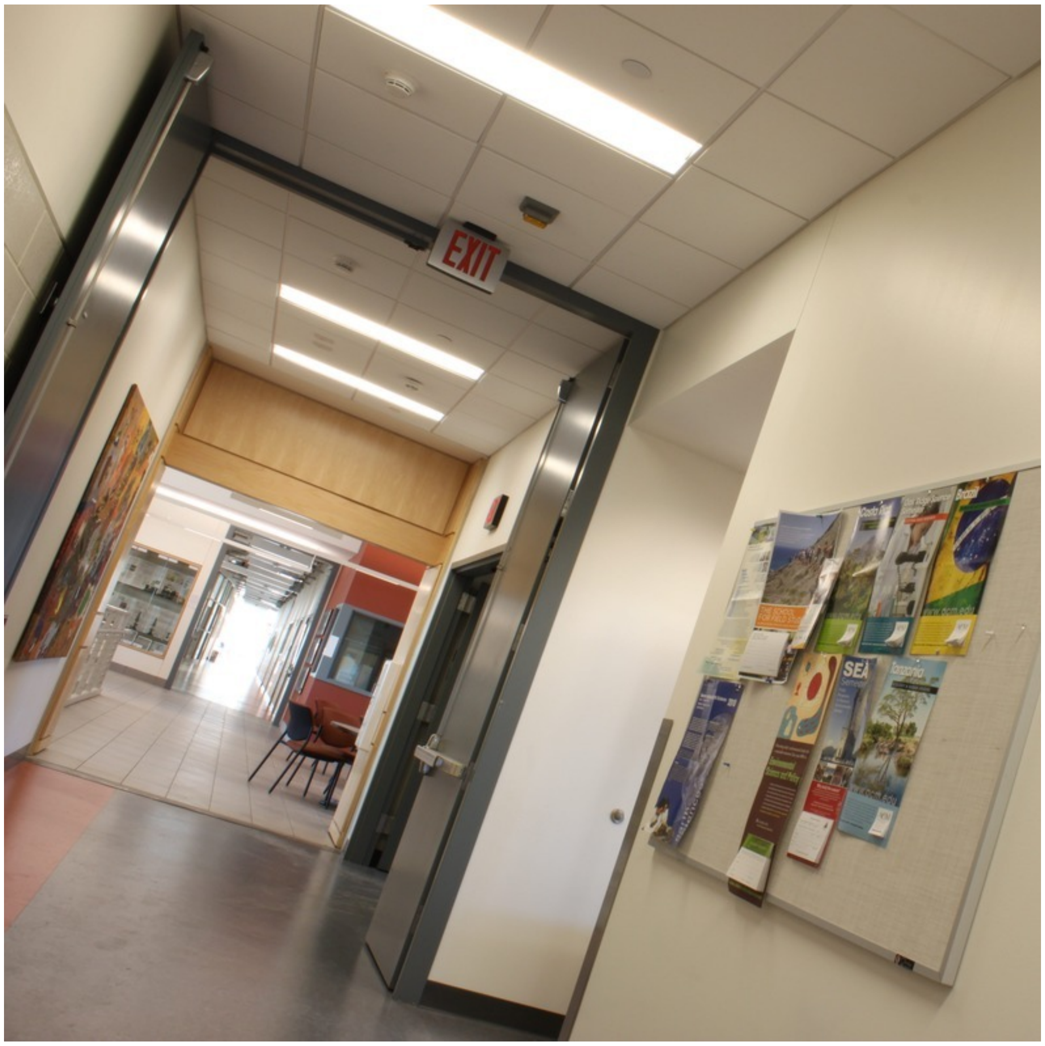
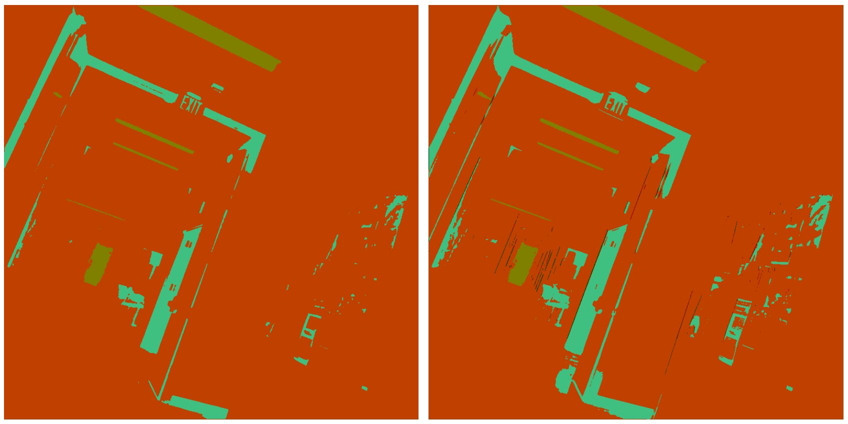
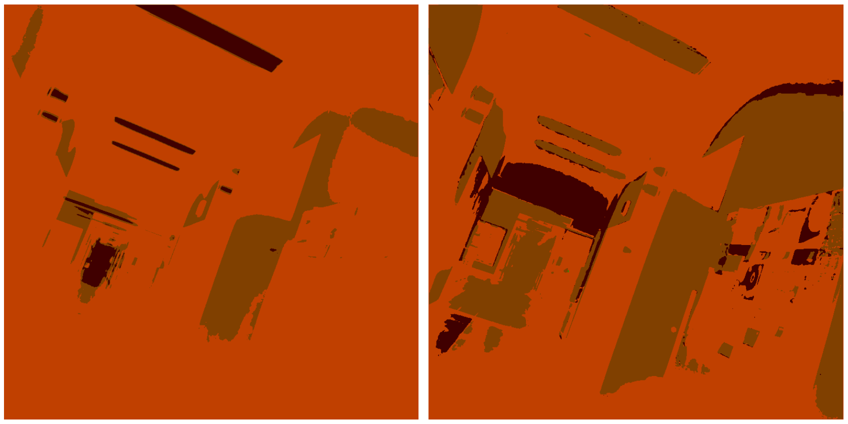


Figure 4: This is the original image that we used for inference results.



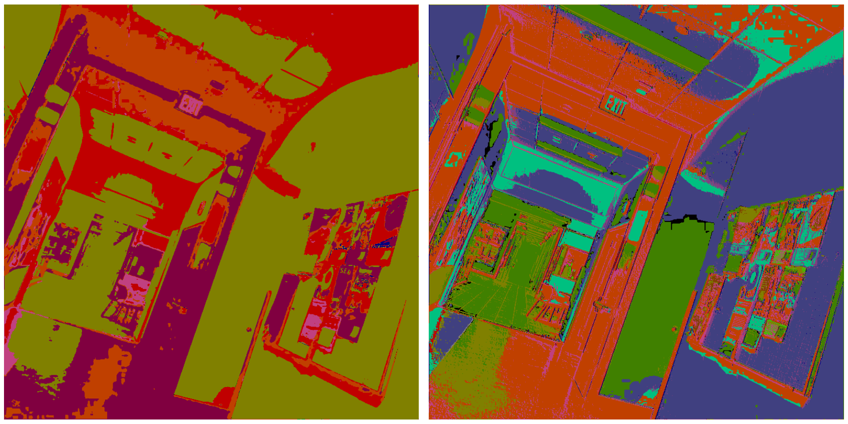
(a) 20_20 without ReLU (b) 20_20 with ReLU

Figure 5: This is the comparison of 20×20 resize-dimension results between adding batch normalization and additional ReLU.



(a) 100_100 without ReLU (b) 100_100 with ReLU

Figure 6: This is the comparison of 100×100 resize-dimension results between adding batch normalization and additional ReLU.



(a) 500_500 without ReLU (b) 500_500 with ReLU

Figure 7: This is the comparison of 500×500 resize-dimension results between adding batch normalization and additional ReLU.

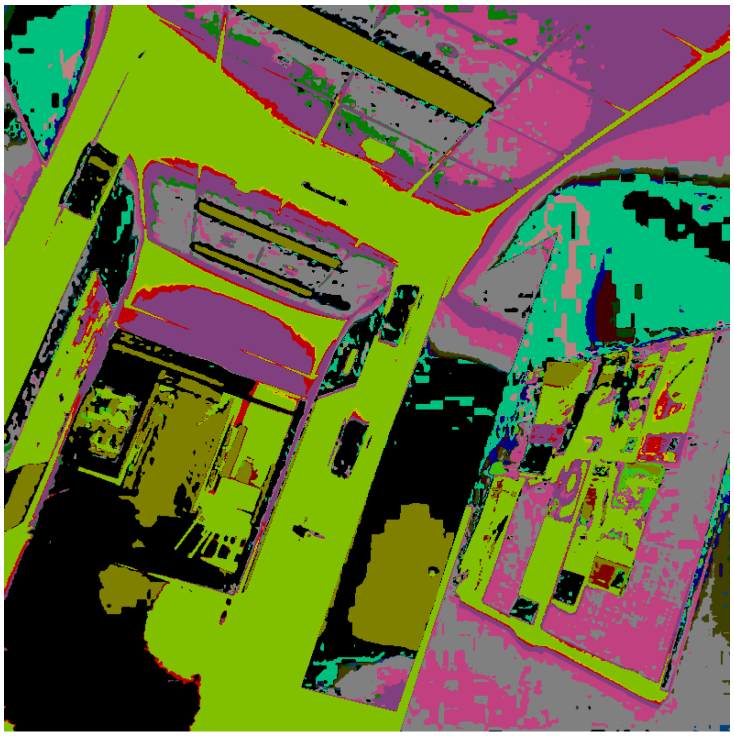
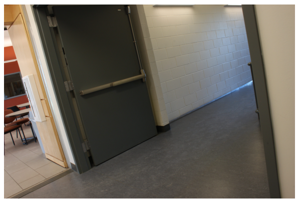


Figure 8: This is the output without batch normalization


(a) Original image

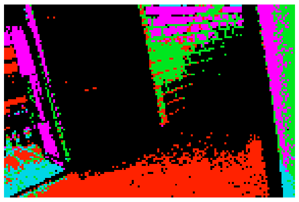
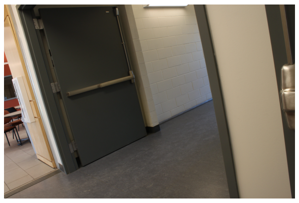

(b) Semantic segmentation

Figure 9: This is the segmentation of the image using the first method.


(a) Original image

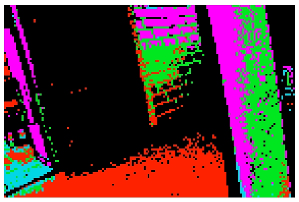
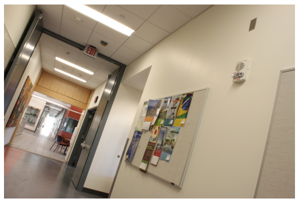

(b) Semantic segmentation

Figure 10: This is the segmentation of the image using the first method.


(a) Original image

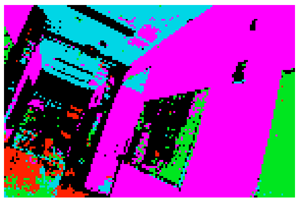
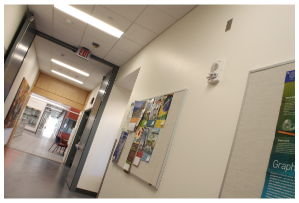

(b) Semantic segmentation

Figure 11: This is the segmentation of the image using the first method.


(a) Original image

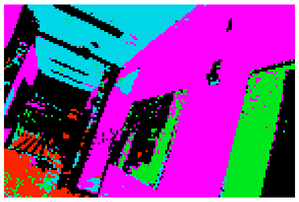
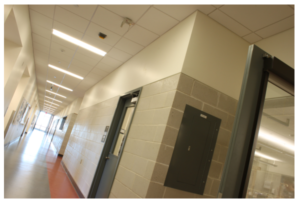

(b) Semantic segmentation

Figure 12: This is the segmentation of the image using the first method.


(a) Original image

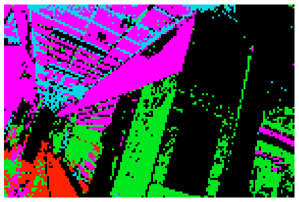
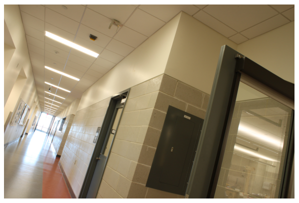

(b) Semantic segmentation

Figure 13: This is the segmentation of the image using the first method.


(a) Original image

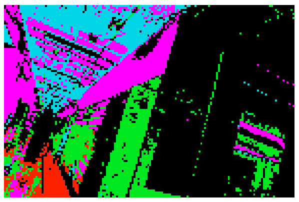
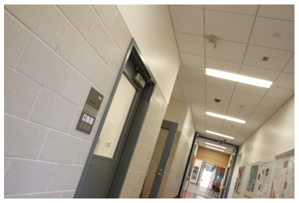

(b) Semantic segmentation

Figure 14: This is the segmentation of the image using the first method.


(a) Original image

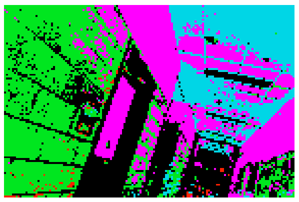
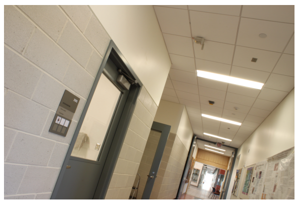


Figure 15: This is the segmentation of the image using the first method.


(a) Original image

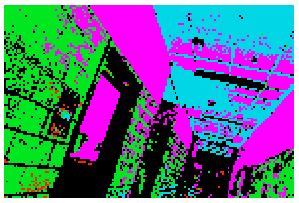

(b) Semantic segmentation

Figure 16: This is the segmentation of the image using the first method.