Indoor Scene Segmentation using Conditional Random Fields

Colin Wei and Helen Jiang
Stanford University
{colinwei, helennn}@stanford.edu

Abstract

Indoor scene segmentation is a problem that has become very popular in the field of computer vision with applications that include robotics, medical imaging, home remodeling, and video surveillance. This problem proves even more difficult when the scene is cluttered. Our project aims to explore ways to improve indoor scene segmentation algorithms by examining and evaluating a popular method.

We focus on evaluating the robustness of the algorithm for indoor scene segmentation described in [16] by Silberman et. al which uses SIFT features and conditional random fields to produce segmentations. In our project, we reimplement their method and compare performance by modifying specific sections.

We find that the neural network used in [16] is not robust to an increasing number of classes, but the CRF model is in the sense that as we increase the number of classes, the CRF becomes increasingly important to produce an accurate segmentation. Furthermore, we also demonstrate that changing the algorithm we use to generate superpixel segmentations increases classification accuracy of the entire pipeline.

1. Introduction

In our paper, we explore the semantic segmentation of indoor scenes, even cluttered ones. The main goal is that given an RGB or RGB-D image of a cluttered indoor scene, we output a properly labeled image with each individual pixel corresponding to an object class, such as a television, chair, or table. Although semantically segmenting a scene is an easy task for humans, automatic segmentation using machines proves to be a challenging problem.

The solution we are implementing is modeled after the segmentation algorithm described in [16], which utilizes neural networks trained on SIFT features along with conditional random fields in order to produce a semantic segmentation.

The goal of our paper is to thoroughly evaluate the algorithm described in [16] in order to understand its success cases and failure cases. We hope that in doing so, we can gain intuition on directions for future work. We follow and re-implement the technical details of Silberman et. al's method of indoor semantic segmentation. Furthermore, we test modifications to the method.

2. Related Work and Contributions

2.1. Literature Review

There is a large body of work on using conditional random fields (CRF's) in order to produce semantic segmentations of images. In [11], He et. al present an approach using multi-scale conditional random fields for image segmentation. They leverage 3 different probabilistic models: a classifier relying only on local information, a conditional random field relying on hidden regional variables to model interactions between object classes, and a CRF that incorporates global label information. Their model relied on a complicated training loop. Subsequent works such as [15, 18] use pairwise potentials to model the interactions between neighboring pixels when producing semantic segmentations. In [16, 14], Silberman et. al follow the same CRF framework and also introduce the NYU dataset, a densely labeled dataset of indoor scenes. Finally, in [8], Chen et. al introduce a state-of-the-art segmentation pipeline which utilizes a deep convolutional neural network and fully-connected CRF to produce accurate segmentations.

2.2. Our Contribution

We reimplement the semantic segmentation approach in [16], and run the following main experiments:

1. We evaluate the ability of the neural network used in [16] to learn 100 object classes. In [16], 13 object classes are used. This experiment allows us to measure the robustness of their neural network.
2. We evaluate the performance of their CRF pipeline using different superpixel algorithms to create initial low-level segmentations. We also qualitatively analyze the performance of the CRF on images in the test set and note potential areas for improvement.

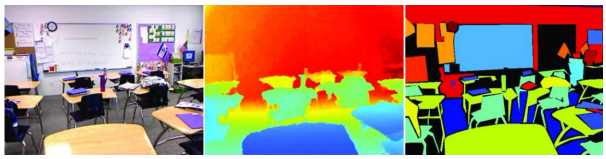


Figure 1: Example data from the NYU dataset: Left=RGB Image; Middle=Raw Depth Image; Right=ground truth class labels created by Amazon Turk

3. We evaluate the performance of their CRF pipeline in the 100 object class setting, and show that the CRF model is robust to an increasing number of object classes.

3. Technical Details

3.1. Dataset

We use the SUN RGB-D dataset [17], which contains the RGB-D images from the NYU depth dataset [14] which is the one that Silberman et. al. created and used, the Berkeley B3D0 dataset [12], and the SUN3D dataset [21].

In our project, we use 1449 images from only the NYU v2 dataset [14], although we use the labels provided by the SUN dataset [17]. The NYU dataset includes the raw RGB images, the raw depth images, and the labeled images, as shown by example in Figure 1. We chose to use the SUN RGB-D version of the NYU images because the SUN RGB-D dataset is around 10000 images total, allowing us to extend our work to settings with more data in the future. The SUN RGB-D dataset provides object class labels for each individual pixel of every image. Although the dataset provides depth information for each image, we only use RGB information for all of our implementations.

3.2. Segmentation Pipeline

To produce a semantic segmentation of an image, we take the following steps, following the method of Silberman et. al:

1. Extract SIFT features $\{f_{ij}\}$ from a dense grid on the image using a sliding window. As in [16], we use a grid with a stride of 10, where each sliding window is of size 40 by 40. Although Silberman et. al concatenate SIFT features from three different scales, we only use one scale for our project in the interest of saving time and computational power.

2. Use a neural network to produce class probabilities $P(\cdot|f_i)$ for each feature location i in the image. As in [16], our neural network has a single hidden layer of size 1000.

3. Use a low-level segmentation algorithm to segment the image into superpixels. We experiment with Felzenszwalb segmentation, the approach used in [16], as well as quickshift and SLIC segmentation. All points i in a superpixel are assigned class probabilities P_i that are equal to the average of the class probabilities for all descriptors corresponding to grid points in a superpixel. If no grid points fall inside a superpixel, we assign the superpixel uniform class probabilities.

4. Model pixel labels as a conditional random field. The energy of the CRF is defined as follows:

$$E(y) = \sum_{i \in I} \phi_i(y_i) + \sum_{i \in I} \sum_{j \in N(i)} \psi_{ij}(y_i, y_j)$$

The summations are taken over all pixels in the image. The first summation models unary potentials for each class, while the second summation models pairwise interactions between neighboring pixels.

Silberman et. al model ϕ_i as the negative log of the product of $P_i(y_i)$ and a location prior on the class y_i. We did not implement location priors and instead set $\phi_i(y_i) = -\log P_i(y_i)$.

Finally, we set pairwise potentials

$$\psi_{ij}(y_i, y_j) = \mathbb{1}(y_i \neq y_j)\eta e^{-\alpha \|I_i - I_j\|_2^2}$$

where I_i and I_j are the i-th and j-th RGB color channels and η, α are hyperparameters. This potential function mirrors the one used in [10]. Although Silberman et. al use a different pairwise potential function, we find that this potential function is easier to tune.

5. Minimize the energy function of the CRF. In [16], Silberman et. al use the scheme provided by [4]. We experiment with both Boykov et. al's α-expansion algorithm in [4] and simulated annealing [2].

3.3. Low-level Segmentation Methods

In our semantic segmentation framework, we utilize three different unsupervised algorithms for the purpose of producing low-level segmentations:

3.3.1 Felzenszwalb [9]

This algorithm is used by Silberman et. al and preserves detail in low-variability regions in the image rather than high-variability regions. The algorithm proceeds by using a graph-based representation of the image to find region boundaries, sequentially combining different regions based on a similarity score across regions.

3.3.2 Quickshift [20]

Quickshift is a mode-seeking clustering algorithm that builds off of variants of mean-shift. The algorithm provides

a density estimate for each pixel location; it then constructs a tree where images are connected to their nearest neighbors with larger pixel values. By cutting branches of the tree whose distance cross a max threshold, we obtain clusters for the image.

3.3.3 SLIC (Simple Linear Iterative Clustering) [1]

SLIC is a clustering method that essentially applies the k-means algorithm on a different feature space, which takes into account pixel location and color intensity. After constructing this feature space, SLIC uses Lloyd's algorithm to output cluster assignments.

3.4. CRF Optimization

Although the minimization of the energy function of a CRF is an NP-hard problem, the α-expansion algorithm [4] and simulated annealing provide general methods for performing this optimization [2].

3.4.1 α-expansion Algorithm

This algorithm by [4] formulates the problem of minimizing the energy function of a CRF as a min-cut problem. Although this problem is still NP-hard in the worst case, Boykov et. al provide an approximation algorithm that finds a local minimum with respect to α-expansion moves, which consists of moves where pixels change their labels to α for a given label α. Given a labeling y and label α, we can compute

$$\min_{y' \in N_\alpha(y)} E(y')$$

where $N_\alpha(y)$ is the set of labels within one α-expansion of y and $E(y')$ is the energy of the labeling y', in polynomial time using a reduction to standard min-cut. The algorithm proceeds by iteratively performing this local minimization procedure.

3.4.2 Simulated Annealing

Simulated annealing is a black-box combinatorial optimization problem that is guaranteed to converge to the optimal solution, though it may be very slow in practice [2]. The algorithm proceeds by sampling local changes to a label assignment (i.e. the configuration of a single pixel at location i) with conditional probability equal to

$$P(y_i^{new}|y) = \frac{\exp\left(-\frac{1}{T}E(y^{new})\right)}{\sum_{y' \in N_i(y)} \exp\left(-\frac{1}{T}E(y')\right)}$$

where $N_i(y)$ is the set of labelings obtained by changing only pixel i, and T is a changing temperature parameter. The idea is to gradually reduce T from high values, where the probability distribution is near uniform, to low values near 0, where the distribution concentrates on labelings that minimize the energy function.

Table 1: Statistics for 11 Class Train/Test Set

Class Name	# Training Descriptors	# Test	Index
Bed	31791	39161	0
Bookshelf	58853	38289	1
Cabinet	99732	80484	2
Ceiling	14025	11884	3
Floor	108333	80595	4
Picture	24932	25235	5
Sofa	34744	47957	6
Table	29964	28882	7
Television	7126	12905	8
Wall	373894	311308	9
Window	30695	28160	10

The total number of training and test examples for each class for our 11 class model.

3.5. Our Implementation Details

We implement everything using Python. To extract SIFT features, we use OpenCV's Python wrapper [5]. To train the neural network over the dense grid of SIFT features, we use the Python package scikit-neuralnetwork [7]. We use the package scikit-image to run the superpixel segmentation algorithms [19]. Finally, for the graph-cut optimization algorithm of [4], we use gco-python, Andreas Mueller's Python wrappers for the gco optimization package [4, 13, 3]. We implemented simulated annealing from scratch using Cython to optimize for speed. We did not use any code from [16], as we implemented all of the preprocessing steps ourselves.

4. Experimental Setup and Results

4.1. Neural Network

We train and test a neural network for classifying image patches. We partition all images from the NYU dataset into the train/test splits provided in [14]. For each training image, we extract SIFT features from the dense grid described in Section 3.2. Likewise, we construct a test set by extracting SIFT features from each image in the test set using the same dense grid. After consolidating training features, we randomly partition the SIFT descriptors into 12 total subsets so we do not have to load the entire training set into memory. One training loop consists of a pass over all 12 training subsets, starting with a learning rate of 0.1 and multiplying by a decay rate of 0.75 with each subsequent subset. Every other hyperparameter for our neural network is set to the scikit-neuralnetwork default.

Because the SUN RGB-D dataset contains 894 different

Table 2: Train/Test Accuracy on 11 Class Set

% Train Correct	% Test Correct
62.06%	49.96%

Accuracy on 11 class training and test sets. Computed for all grid points for each train/test image.

Table 3: Train/Test Accuracy on 100 Class Set

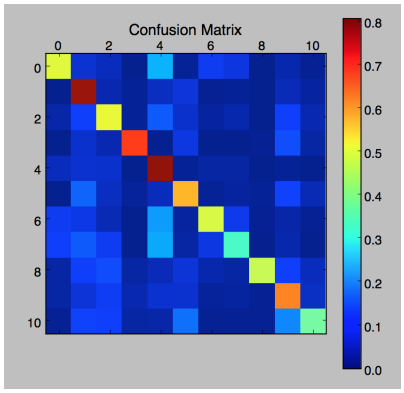
% Train Correct	% Test Correct
25.89%	13.37%

Accuracy on 100 class training and test sets. Computed for all grid points for each train/test image.

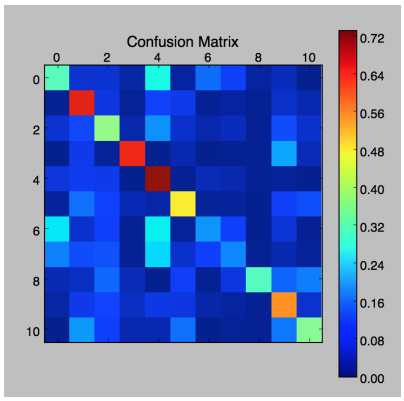
object class labels, a number that is too big for a neural network with a single hidden layer to accurately classify, we instead train on subsets of the classes. In total, we train 2 different neural networks. Following [16], we handpick 11 object classes to train on, shown in Table 1. Silberman et. al use 13 classes - the classes that we use, in addition to the "blind" and "background" classes. We do not use these classes, however, because they do not appear in the SUN RGB-D labels. We also choose the 100 most common labels, and we train a neural network to classify between these classes. For reference, our code includes a pickle file containing the names of these 100 classes. The 11 handpicked classes form a large subset of these 100 most common classes.

From Table 1, we can see that class distributions between train and test are pretty similar, but class distributions are both very skewed. This also holds for the 100 classes set. Because of this skewed distribution, it is possible to train a neural network that achieves high test accuracy but only learns a few classes properly. To remedy this issue, we balance the training distribution by capping each class at 5000 examples per split in the 11 class case and 1000 examples per split in the 100 class case. We are unclear on how Silberman et. al work around this problem.

In Table 2 and Figure 2, we show the accuracy results and confusion matrices for the 11 class dataset. In Table 3 and Figure 3, we show the accuracy results and confusion matrices for the 100 class dataset. From the discrepancy between the training and testing accuracy for both datasets, it is clear that our models overfit, even though we use a substantial amount of training data given the size of networks we train. There are two main reasons why this could happen. First, descriptors from the same image corresponding to nearby points possess some redundancy, which means the effective number of training samples is smaller than the actual number. Second, intra-class discrepancy is very high between different indoor scenes. Since the train/test splits in [14] are arranged so that not a single scene is in both train and test, test images could present variations of a class not



(a) Confusion matrix for training.



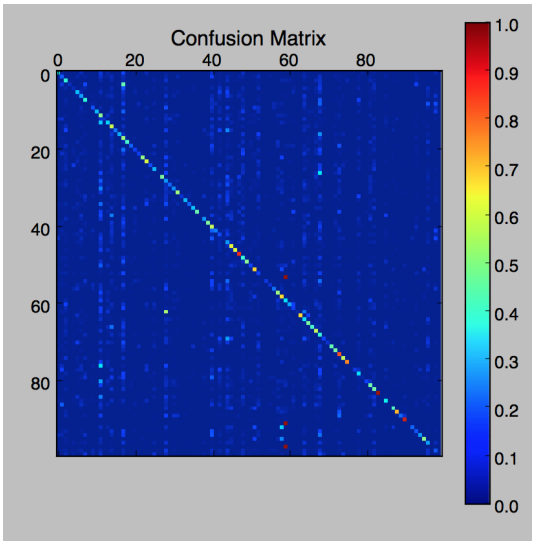
(b) Confusion matrix for testing.

Figure 2: Confusion matrix for 11 class dataset. Indices correspond to indices in Table 1.

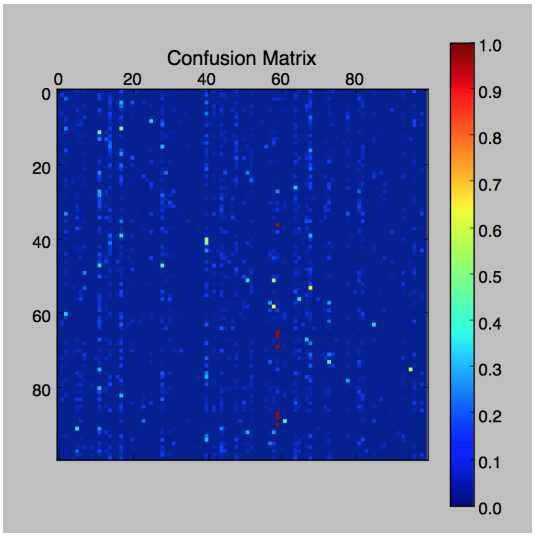
seen during training.

Judging from the confusion matrices in Figure 2, it seems that the 11 class neural network performs worst on classes that are both relatively scarce and similar in appearance to other classes. For example, sofas and tables are commonly classified as floors. These sofa and table classes are very scarce compared to floors and exhibit similar properties such as a large flat surface, leading to this incorrect classification. We have already mitigated many of these incorrect classifications by trying to balance classes during training, but we are unsure how to improve this further without switching to a deeper network architecture. Since Silberman et. al do not provide their neural network results in [16], we cannot perform a direct comparison. However, their CRF with only unary potentials achieves a 40.9% pixel accuracy on 13 classes, which implies that our results on 2 fewer classes are on a comparable performance level.

We cannot make any comparison to [16] on the 100 class case because they only provide results for 13 classes. However, the confusion matrix in Figure 3 shows that the single layer neural network is not robust enough for the 100 class case. Many of the classes that are very incorrectly



(a) Confusion matrix for training.



(b) Confusion matrix for testing.

Figure 3: Confusion matrix for 100 class dataset. Indices are available in the code.

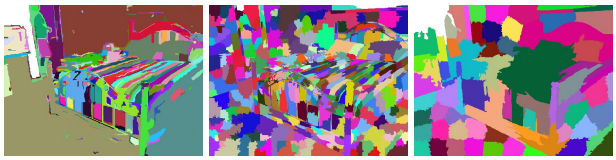
classified in the test confusion matrix are also very uncommon; for example the class "backpack" (index 8) appears only 539 times throughout the entire training set, and the model rarely outputs that label. An interesting question is whether the model fails because SIFT descriptors cannot capture enough information about all 100 classes or because the network architecture itself is flawed. In future work, we could investigate this by training the single layer network on the raw image patches and observing whether this improves results, which would imply SIFT as the problem. If the results do not improve, a deeper network may be necessary.

Table 4: Superpixel Algorithm and 11 Class CRF Performance

Superpixel Alg.	Unary Acc.	CRF Acc.
Felzenszwalb	48.74%	49.95%
Quickshift	48.20%	51.21%
SLIC	49.93%	51.35%

Accuracy of pixel-level labels for segmentation of test images on 11 classes. We only consider pixels that fall in one of the 11 classes. Unary accuracy is computed from the segmentation given by minimizing the unary terms of the energy function. CRF accuracy is computed from considering pairwise terms too.

(a) Original image



(b) Felzenszwalb, quickshift, SLIC superpixels

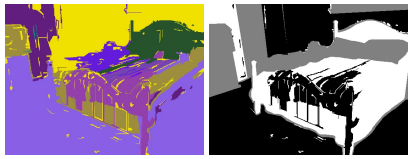
Figure 4: Example superpixel segmentation.

4.2. Superpixel Algorithms and CRF Performance

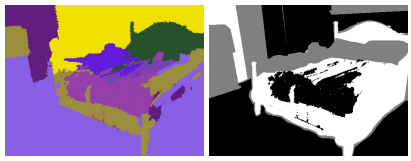
For the 11 class case, we analyze the performance of the entire segmentation pipeline described in Section 3.2, varying the algorithm we use to create initial superpixel segmentations. For the scikit-image implementation of the Felzenszwalb algorithm, we set the scale parameter to 100 based on qualitative evaluation of a few training images. For quickshift and SLIC, we use the default hyperparameters from the implementation of scikit-image.

Table 4 shows the accuracy results for our different trials over the entire test set. To produce the results in Table 4, we use Boykov et. al's α-expansion algorithm for minimizing the CRF energy for each trial [4, 13, 3]. We use 5 iterations for the α-expansion algorithm.

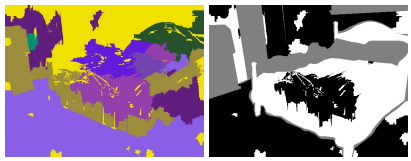
Figure 5: The images on the left show segmentation results for different superpixel initializations in Figure 4. On the right, the truth map is show. Black means the model classified the pixel correct, white means the model classified incorrect, and gray means the pixel does not belong to any of the 11 classes.



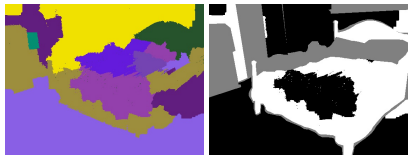
(a) Felzenszwalb unary



(b) Felzenszwalb CRF



(c) Quickshift unary



(d) Quickshift CRF

(e) SLIC unary

(f) SLIC CRF

Surprisingly, we find that performing Felzenszwalb segmentation to create superpixels, the method that Silberman et. al use in [16], actually results in the worst performance out of all the superpixel segmentation methods that we try. The accuracy rate of the conditional random field, 49.95%, is more or less the same as the test accuracy of the 11 class set, and the accuracy of the unary model with Felzenszwalb is worse. We should note that the two test sets are different - the test set in Section 4.1 uses only the subset of pixel locations along the dense grid, while our test set here uses all pixels. Even so, it seems that first performing the superpixel segmentation with Felzenszwalb and quickshift actually hurts the performance of the classifier. This is because, as seen from Figure 5, both Felzenszwalb and quickshift create tiny clusters that do not contain any grid points (see the tiny yellow clusters in Figures 5a and 5c) and are thus assigned uniform superpixel probabilities. SLIC works well even restricted to unary potentials because it produces larger clusters, ensuring that a grid point falls in each cluster.

In the CRF setting with pairwise potentials, quickshift and SLIC both provide better performance than Felzenszwalb segmentation. This is because, as seen in 4b, quickshift and SLIC provide more balanced cluster sizes, whereas Felzenszwalb segmentation can often produce very large superpixels. Since all pixels within a superpixel share the same class probabilities, if these large superpixels have the wrong class probabilities, performance will suffer. Meanwhile, for quickshift and SLIC superpixels which are smaller, as long as a few superpixels have the correct class probabilities, the CRF will be able to fix the labels for adjacent superpixels too.

For all three superpixel algorithms, the CRF is able to correct some of the mistakes seen in the unary version of the model. For example, it can fix nearly all of the inaccuracies introduced by superpixel clusters that are too small, as seen in the smoothness of the CRF segmentations in Figure 5. Furthermore, the CRF is able to produce minor improvements in fixing some small wrongly classified regions. The potential for improvement is limited in three ways:

1. The accuracy of the neural network is too low. If the neural network is too confident in the wrong labels, then it is hard for the CRF to change as desired. In future work, we can examine whether using a CRF with more powerful neural network models results in more performance gains solely from the CRF.

2. The spatial transition potentials do not provide a good enough model of the transitions between classes. As an example, this occurs in Figure 5f. The CRF is able to correctly label more parts of the floor near the right side of the bed. As seen in Figure 4b, a large chunk of the patch of the floor incorrectly classified in the SLIC unary belongs to the same superpixel. Since the CRF

Table 5: 100 Class CRF Performance

Superpixel Alg.	Unary Acc.	CRF Acc.
Felzenszwalb	15.16%	15.78%
SLIC	14.77%	15.67%

Accuracy of pixel-level labels for segmentation of test images on 100 classes. We only consider pixels that fall in one of the 100 classes. Unary accuracy is computed from the segmentation given by minimizing the unary terms of the energy function. CRF accuracy is computed from considering pairwise terms too.

is able to fix half of this superpixel, neural network inaccuracy is not the only problem - therefore, transition potentials must be an issue too.

3. The CRF can fix labels for pixels only if labels for nearby pixels in the same object are correct. This is because the CRF model is based on pairwise interactions between neighboring pixels. In future work, we could attempt to fix this issue by using a fully connected CRF model as in [8], which allows the model to account for global feature interactions.

Another way to potentially address the first and second limitations is to fine-tune the neural network probabilities and learn the pairwise interaction potentials by directly training a CRF model instead of handcrafting the pairwise potentials. We can do this by modeling the pairwise potentials as the result of some convolution kernel applied to the local image patch followed by some nonlinearity. We could formulate a training objective that maximizes the log-likelihood of the true labels, optimize it using contrastive divergence [6], and perform back-propagation into the unary and pairwise potentials to learn these functions. We suspect that this approach could mitigate the first and second limitations by providing an energy function that is optimized for the desired task, producing correct pixel labels. Because of lack of computational power, we leave this idea for future work.

4.3. Varying the Number of Classes

We also explore the robustness of the approach in [16] to a varying number of classes. To do so, we run the segmentation pipeline in Section 3.2 the test set using the 100 most common class labels. We run our experiments using Felzenszwalb segmentation and SLIC in order to generate superpixels; we forgo testing quickshift because it provided similar performance to SLIC in our 11 class test set and the segmentation algorithm takes too long to run in conjunction with performing α-expansion optimization on 100 classes, which already requires significantly more time than the 11 class case.

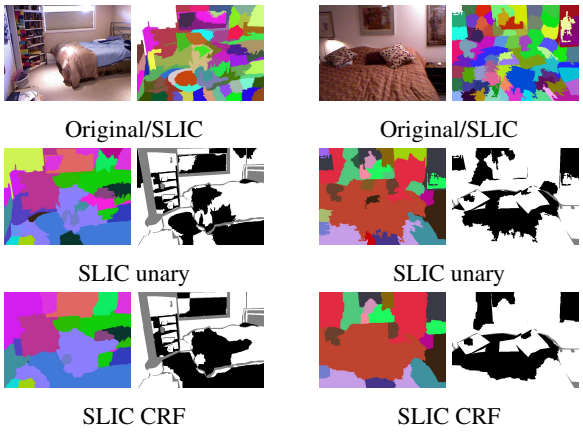


Figure 6: Sample truth maps and segmentations for the 100 class case.

We provide our accuracy results in Table 5. Surprisingly, whereas the superpixel segmentations hurt our unary potential performance in the 11 class case, they actually improve our performance in the 100 class case over train and test accuracy given in Table 3. We are unsure why this is the case. In addition, Felzenszwalb actually provides better performance for unary potentials now. This discrepancy might be due to the fact that in the 100 class case, larger clusters might result in significant improvement for some test images because averaging class probabilities of large clusters reduces noise, and there is more noise in the 100 class case as opposed to the 11 class case.

The increase in accuracy percentage between the unary and CRF models is an interesting result that demonstrates the robustness of the CRF model. As the number of classes increases, the CRF actually seems to make a bigger impact on the final segmentation. Although the actual increase in accuracy is lower in the 100 class case than the 11 class case, the increase is higher in proportion because many fewer pixels get labeled correctly in the 100 class case.

From the examples shown in Figure 6, we can also qualitatively observe the increased impact of the CRF on segmentation quality. In Figure 6, we show example segmentations using the SLIC superpixel algorithm; we choose to analyze SLIC because SLIC and Felzenszwalb exhibit similar CRF performance on the 100 class set, while SLIC is clearly better on the 11 class set. Whereas the truth maps in Figure 5 do not change much between the unary and pairwise cases, the truth maps shown in Figure 6 exhibit significant changes between the two cases. Furthermore, the segmentations obtained using the CRF contain far fewer clusters than the segmentations obtained only using the unary potentials.

We can explain the increased impact of the CRF as follows: since there are a larger number of classes the network is less certain about its classification choices and therefore assigns more uniform class probabilities. As a result, the pairwise potential term is larger in magnitude than the unary

Table 6: Optimization Algorithm Comparison

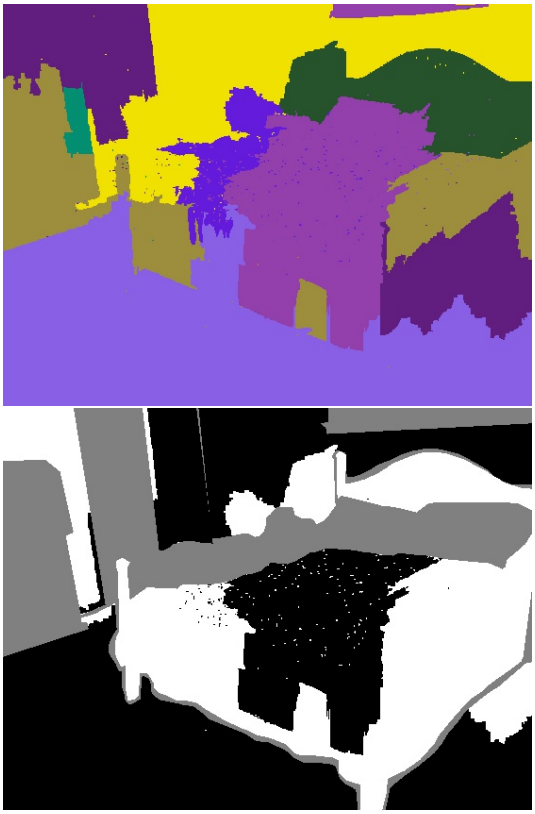
Optimization Alg.	% Acc on Random Sample
Simulated Annealing	51.55%
Boykov et. al	53.01%
Unary	51.56%

Accuracy of pixel-level labels for segmentation on 11 classes using different optimization scheme. In the "Unary" scheme, we ignore pairwise potentials and take the argmax class probability for each pixel. For all optimization methods, we use SLIC to obtain superpixels.

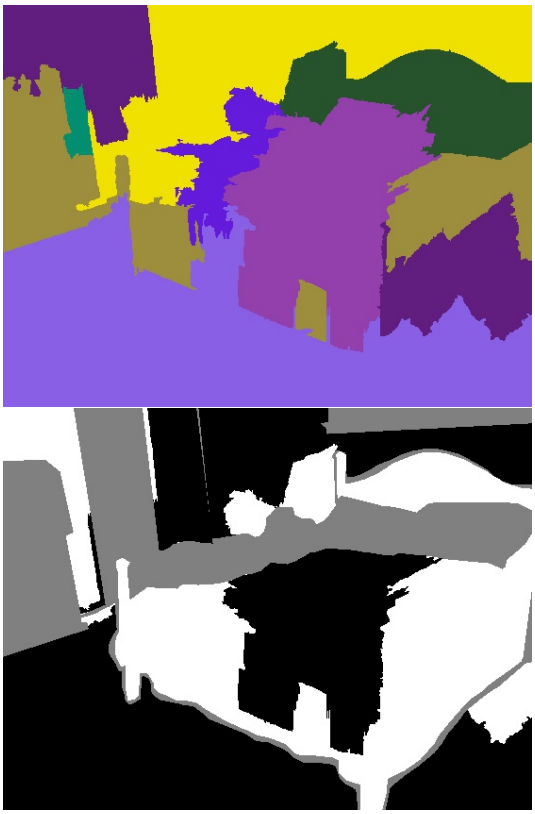
term, which means the decisions resulting from pairwise potentials are weighted more heavily. Thus, we can conclude that while the neural network model is not robust to an increase in the number of classes, the CRF is quite robust. Thus, perhaps we can expect further improvements by using better CRF potentials as described at the end of Section 4.2. One interesting question for future work is whether the CRF would still make as big of an impact in the case where we have a better neural network model for classifying the grid points.

4.4. Simulated Annealing

In an attempt to determine whether the optimization algorithm used for the CRF is optimal, we implement simulated annealing, a black box algorithm useful for Markov Random Fields [2]. We anneal from a temperature of 1 to 0.01, using 15 total linearly spaced temperatures. Each annealing step, we perform a Gibbs sampling sweep over every single pixel in the image, sampling new class labels with probability proportional to $\exp\left(-\frac{1}{T}E(y)\right)$, where T is the current temperature and $E(y)$ is the energy of CRF on labels y. We initialize our labels using the minimum energy configuration for unary potentials - although we tried different initialization methods, we found that this worked best empirically. Because we sweep every single pixel in the image every Gibbs sampling step, our implementation is extremely slow, even when we use Cython. Therefore, we only run our segmentation pipeline on a random sample of 69 test images, using only the 11 class model. For fairness, we compare to the α-expansion algorithm in [4] and unary potentials using the same random sample. We show results in Table 6.



(a) Simulated annealing with SLIC



(b) Unary with SLIC

Figure 7: Example segmentation using simulated annealing.

Unfortunately, simulated annealing does not seem to optimize the CRF energy at all. As shown in our example, which is the same initial image as used in Figure 4 and Figure 5, our simulated annealing implementation does very little to change the initial assignment. All examples in our test set appear similar to this. This suggests that some of our hyperparameters are not set correctly. For example, our initial temperature might be too high, resulting in the Gibbs sampler getting "stuck". Furthermore, we might not have run a sufficient number of iterations of simulated annealing. Given more time, we could try to optimize these hyperparameters to obtain better results. However, we also conclude that given the large number of hyperparameters to optimize and the high computational cost, simulated annealing is not worth the effort compared to cut-based segmentation.

5. Conclusion and Future Work

In this paper, we implement Silberman et. al's image segmentation pipeline in [16] with the final goal of exploring the strengths and weaknesses of their algorithm. We show that the neural network model based on SIFT features works sufficiently well for a small number of classes, but does not work for classification tasks on a larger number of object classes. We also analyze and explain the performance of different superpixel algorithms in place of Felzenszwalb's segmentation algorithm, and we find that SLIC is optimal in terms of both speed and final segmentation quality. Furthermore, we experiment with a larger number of object classes in our test set, and we show that the CRF framework is very robust to an increasing number of classes, even if the neural network model is not. Finally, we compare the performance of Boykov's segmentation algorithm to simulated annealing, and we find that Boykov's algorithm works much better for our problem.

Our hope is that our experimentation provides interesting directions for future research and we believe that our analysis of the CRF's performance does this. In Section 4.2, we discuss limitations of the CRF and argue that directly learning some CRF potentials from data is a promising direction for further work. In Section 4.3, we show that the CRF becomes more important as the number of classes increases. Another interesting direction for future work is to see if this still holds for an even larger number of classes and also differing CRF connectivities.

Our code can be found at:

https://github.com/cwein3/im-seg

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Ssstrunk. Slic superpixels compared to state-of-the-art superpixel methods. 3

[2] D. Bertsimas and J. Tsitsiklis. Simulated annealing. *Statist. Sci.*, 8(1):10–15, 02 1993. 2, 3, 8

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, Sept 2004. 3, 5

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, Nov. 2001. 2, 3, 5, 8

[5] G. Bradski. Opencv. *Dr. Dobb's Journal of Software Tools*, 2000. 3

[6] M. A. Carreira-Perpinan and G. E. Hinton. On contrastive divergence learning. 7

[7] A. Champandard and S. Samothrakis. sknn: Deep neural networks without the learning cliff. 2015. 3

[8] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, abs/1412.7062, 2014. 1, 7

[9] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, Sept. 2004. 2

[10] L. Grady. Random walks for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(11):1768–1783, Nov. 2006. 2

[11] X. He, R. S. Zemel, and M. A. Carreira-Perpinan. Multiscale conditional random fields for image labeling. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–695–II–702 Vol.2, June 2004. 1

[12] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell. A category-level 3-d object dataset: Putting the kinect to work. in iccv workshop on consumer depth cameras for computer vision. 2011. 2

[13] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *Proceedings of the 7th European Conference on Computer Vision-Part III*, ECCV '02, pages 65–81, London, UK, UK, 2002. Springer-Verlag. 3, 5

[14] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 1, 2, 3, 4

[15] J. Shotton, J. Winn, C. Rother, and A. Criminisi. *Computer Vision – ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I*, chapter TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-class Object Recognition and Segmentation, pages 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. 1

[16] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *Proceedings of the International Conference on Computer Vision - Workshop on 3D Representation and Recognition*, 2011. 1, 2, 3, 4, 6, 7, 9

[17] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 2

[18] B. Triggs and J. J. Verbeek. Scene segmentation with crfs learned from partially labeled images. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1553–1560. Curran Associates, Inc., 2008. 1

[19] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014. 3

[20] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. 2008. 2

[21] J. Xiao, A. Owens, and A. Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *ICCV*, 2013. 2