# Road and Building Classification on Satellite Images Using Superpixels and Semi-Supervised Learning

Eric Chu, *MIT Media Lab*

***Abstract*—In order to scale segmentation of aerial satellite images, I took a semi-supervised approach to handle the scarcity of labeled data. Previous work first used SVM's to predict labels at the superpixel level, followed by label propagation between superpixels. For our research, we are interested in segmentation at the pixel level. We thus attemped different methods to perform label propagation and labeling at the pixel level.**

## I. Introduction

With the increased availability in the past decade of high resolution satellite imagery, we now have the ability to use machine learning techniques to automatically segment satellite images into roads, buildings, and more. This can in turn be used for map creation and rural development planning.

My group Social Machines in the Media Lab is working on a project to increase literacy and create more responsive communities in rural villages. We aim to estimate the social structure of these communities in part through satellite imagery of villages in India and Sub-Saharan Africa. With our inferred network, our goal is to enable targeted intervention and optimized distribution of information, education, technologies, goods, and medical aid.

### A. *Related Work, Motivation, and Approach*

Mnih and Hinton have a series of papers that use twenty-layer deep convolutional neural nets (CNNs) to classify each pixel of a satellite image as belonging to a road, building, or other [7]. Using the architecture and dataset released by Mnih, my group has successfully replicated their classifier in Caffe. It takes about two days to train on $\sim$ 130 1500 x1500 pixel images using 3-4 GPUs.

There are two negatives with this approach. First, the provided dataset of Massachusetts roads and buildings has been meticulously labeled. This quantity and quality might not scale. Second, the time to train is longer than we would prefer. This would not be a problem if we could simply train a model once and be done. However, the variance in building and town shapes, sizes, and colors dictates we will need to train a model per region.

The scarcity of labeled data suggests exploring a semi-supervised approach. Sethi et. al uses such an approach to classify neighborhoods from satellite images [8]. They first segment the image into superpixels. They then calculate computer vision features for each superpixel. After labeling $\sim$1% of the superpixels, they use a SVM trained on the superpixel features to create preliminary labels. Next, they create a neighborhood graph of superpixels. They then use this graph, the ground-truth 1% of labels, and the preliminary labels produced by the SVMs for label propagation to produce the final labels.

I aimed to replicate the majority of the above paper's pipeline to classify pixels as Roads, Buildings, and Other. The big difference is that my problem requires labels at the pixel level instead of simply at the superpixel level. I hoped that the initial labels at the superpixel level would be correct enough that I could then do label propagation at the pixel level.

## II. Data Pre-Processing

My dataset consisted of 151 1500 x 1500 pixel satellite images of Massachusetts. Per image, there was another RGB image map that provided the ground truth labels of Road, Building, or Other. See a) in figure 2. I had to remove some images that were corrupted with stray white marks.

To segment the satellite images into mutually exclusive sets of pixels called superpixels, I used the Ultrametric Contour Mapping (UCM) technique, with code from the Berkeley research group. UCM segments an image into superpixels by hierarchically combining contours, which are calculated using brightness, color, and texture gradients. Like other techniques, UCM first creates a neighborhood pixel graph as an input into spectral clustering. However, it differs in its handling of the resulting eigenvectors; by taking Gaussian directional derivative filters, it can properly segment larger sections of the image that are smooth, such as a blue sky. UCM also has a parameter $k$ that controls the granularity of segmentation. Smaller $k$'s results in smaller superpixels. As UCM is hierarchical, decreasing $k$ will subdivide a superpixel into smaller superpixels [2].

The provided MATLAB code has a function that breaks large images into smaller pieces, calculates the contours, and then stitches the pieces back together. Even so, this was intractable for my images. After I parallelized this code, UCM took 9.66 minutes per image on an Nvidia DIGITS machine with 64GB of RAM.

## III. Superpixel Features

After vectorizing all matrix accesses, saving maps from superpixels to pixels, and saving maps from superpixels to labels, creating the features for superpixels at k = 0.001 took 20 seconds to run on a 2.5 GHz Macbook Pro with 16 GB of RAM.

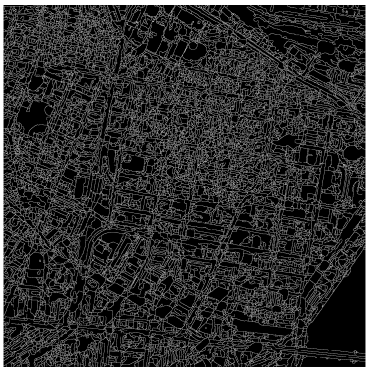
### A. *Computer Vision (CV) Features*

I used the scikit-image library to get three categories of CV features. First, I took the average RGB values of the pixels within the superpixel. Second, I computed the binned greyscale histogram, with each bin being one feature that counts the number of pixels within that intensity range. I hoped this could help capture fine differences, such as shadows in between roads and buildings. Finally, I computed the Harris corner density. Harris corners are found by calculating gradients. If two gradients meet orthogonally, then the point is likely a corner [5]. I tweaked two parameters: the $\sigma$ for the Gaussian filter first applied to the image, and the $minDistance$ between corners. Both serve to avoid classifying noise as corners. The eventual choice was selected after visually inspection showed that it appeared to be catching building corners.

### B. *Parent Superpixel Features*

Because UCM is hierarchical, superpixels at k = 0.1 will be divided into a set of smaller superpixels at k = 0.001. My models were ran on the small k = 0.001 superpixels.

(a) Satellite Image



(b) UCM segmentation

Fig. 1. Sample satellite image and its UCM superpixel segmentation at threshold k=0.5. Note: other images are less urban.

I first computed the size of the parent superpixel (number of pixels), with the intuition that larger, open areas such as a body of water should have a larger size than a road or building. Next, I computed the density of the parent (number of child superpixels), with the idea that parents that contain many buildings would have the highest density.

## IV. Labeling Superpixels

The ground truth labels are at the pixel level, but my models would be trained and tested at the superpixel level. As many superpixels were composed of pixels from different classes, I tried several different labeling methods to address this ambiguity. I refer to these superpixels as **ambiguous superpixels**. Note that the Unanmioius and Min-Ratio methods do not label every superpixel.

1) **Unanimous**: The model was trained and tested only on superpixels in which all pixels have the same label. The superpixel received the same label.

2) **Min-Ratio-$n$**: The model was trained and tested only on superpixels in which at least $n\%$ of the pixels shared the same label. The superpixel received this label.

3) **Majority**: All superpixels would be labeled, with each receiving the label that the majority of its pixels had.

4) **Confidence-$n$**: Each road, building, and other was split into 3 classes, for a total of 9 classes. For instance, superpixels in which all pixels were Roads would be class 0. Superpixels in which at least $n\%$ (but not all) of the pixels were Roads would be class 1. Superpixels in which the majority of pixels were Roads would be class 2.

## V. Models on Superpixels

The SVM we coded for HW2 used the CVXOPT quadratic programming solver. This is only tractable for smaller datasets in which the kernel matrix can stored in memory. Thus, I implemented the SMO algorithm and tested on the HW2 datasets. While its run-time of $\Omega(n^2d)$ is faster than QP (inversion of the kernel matrix is $O(n^3)$), where n is the number of training points and d is the dimensionality of the points, it was still intractable for my large dataset. Therefore, I finally used the scikit-learn library of linear classifiers that are solved using stochastic gradient descent (SGD). As SGD was used, I made sure to use feature scaling.

### A. *Support Vector Machines (SVM)*

#### 1) *Sequential Minimal Optimization (SMO)*

SMO is a coordinate descent approach for optimizing the dual form of the SVM [9]. Recall that the dual formulation of Hard SVM is as follows:

$$\begin{aligned} max_\alpha \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)} y((i)\rangle \\ \text{s.t. } 0 \leq \alpha_i \leq C, i = 1, ..., m \\ \sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \end{aligned} \tag{1}$$

At each iteration of SMO, we select two $\alpha$'s and test if the KTT conditions are satisfied.

$$\begin{aligned} \alpha_i = 0 \Rightarrow y^{(i)}(w^T x^{(i)} + b) \geq 1 \\ \alpha_i = C \Rightarrow y^{(i)}(w^T x^{(i)} + b) \leq 1 \\ 0 \leq \alpha_i \leq C \Rightarrow y^{(i)}(w^T x^{(i)} + b) = 1 \end{aligned} \tag{2}$$

We cannot optimize one $\alpha$ at a time; from the last constraint in (1), holding all other $\alpha$'s constant determines the last $\alpha$. Therefore, we instead optimize two $\alpha$'s at a time.

$$\alpha_1 y^{(1)} + \alpha_2 y^{(2)} = -\sum_{i=3}^{m} \alpha_i y^{(i)} = \zeta \tag{3}$$

Our problem now becomes a two variable optimization problem. By using (3) to write $\alpha_1$ as a function of $\alpha_2$, we can plug the value back in to get a problem quadratic in $\alpha_2$. Then we can take the derivative and set to zero. The update steps for $\alpha$ are given by the following equation (update step for b omitted for brevity's sake):

$$\begin{aligned} \alpha_j := \alpha_j - \frac{y^{(j)}(E_i - E_j)}{\eta} \\ \text{where } E_k = (w^T(x^{(k)}) + b) - y^{(k)} \\ \eta = 2\langle x^{(i)}, x^{(j)} \rangle - \langle x^{(i)}, x^{(i)} \rangle - \langle x^{(j)}, x^{(j)} \rangle \end{aligned} \tag{4}$$

Note that because $\alpha_1$ and $\alpha_2$ must lie in the box [0,C] x [0,C] and the relationship between the two is linear, there is a further constraint that $L \leq \alpha_2 \leq H$. We use $L$ and $H$ to clip the optimized values of $\alpha_2$ if it lies outside the bounds. Some geometric intuition and algebraic manipulation give the following values for L and H:

$$\begin{aligned} \text{If } y^{(i)} \neq y^{(j)}, L = max(0, \alpha_j - \alpha_i), H = min(C, C + \alpha_j - \alpha_i) \\ \text{If } y^{(i)} = y^{(j)}, L = max(0, \alpha_i + \alpha_j - C), H = min(C, \alpha_i + \alpha_j) \end{aligned} \tag{5}$$

I implemented a modified version of SMO that terminates when $k$ iterations pass without any $\alpha$'s being updated. This means we are no longer guaranteed to find the optimal solution

[9]. In my tests, however, results were comparable to the QP-solver method.

*2) **Stochastic Gradient Descent (SGD)***

SVM's are commonly solved in the dual form, in which the reformulation frames the objective function as a sum of inner products between training examples. These inner products can then be computed as kernels that implicitly map points to a higher-dimensional feature space, in which the points then become linearly separable. However, because we use a linear kernel, we can use SGD to optimize the primal form objective. The primal form (with L2 regularization) is as follows:

$$R_{hinge}(w) = \frac{1}{N}\sum_{i=1}^{N} max\big(0, 1 - y^{(i)} w^T \phi(x^i)\big) + \frac{\lambda}{2}||w||^2 \tag{6}$$

*3) **Multiclass SVM***

Unlike logistic regression, SVM's do not naturally generalize to multi-class. Under the one-vs-rest approach, K SVM's are trained (one per class). In the one-versus-one approach, K(K-1)/2 different binary SVM's are trained (one for each pair); the points are then classified by counting the number of 'votes' across all classifiers. However, note that in both cases, a point may be assigned multiple labels. Following empirical results of different normalization methods found in [10], the scikit-learn implementation of the SGD classifiers uses one-versus-rest with simple normalization, in which the probabilities are simply normalized to sum to one.

### B. Logistic Regression (LR)

For a feature vector $\phi$, binary LR is given by:

$$p(C_1|\phi) = \sigma(w^T\phi) = \frac{1}{1 + e^{-w^T\phi}} \tag{7}$$

This generalizes to multi-class through the soft-max function:

$$p(C_k|\phi) = \sigma(w^T\phi) = \frac{exp(w_k^T\phi)}{\sum_j w_j^T\phi} \tag{8}$$

By formulating LR as (6) but with log loss instead of hinge loss, LR can also be solved using SGD.

### C. Weighted Classes

In SGD, classes can be weighted by treating the weight as a multiplier for the update step. In the dual SVM, class weights correspond to the penalty C's. If we have two separate groups of points, and we care twice as much about classifying group one correctly, then we can simply penalize errors for group one twice as much. The Soft SVM thus becomes (with the appropriate slack constraints not shown):

$$min_{w,\zeta,\xi}\frac{1}{2}||w||^2 + 2C\sum_{i=1}^{N}\zeta_i + C\sum_{j=1}^{M}\xi_j \tag{9}$$

Alternatively, we can just duplicate the points in the first group. In the context of weighted classes, the groups are simply the classes. Ultimately, using different weights didn't improve the results much. I ended up using scikit-learn's balanced weight for my models, in which the weight is inversely proportional to the number of occurrences of the class.

## VI. Semi-Supervised Learning and Label Refinement

Semi-supervised learning (SSL) utilizes both labeled and unlabeled data to train a model. In order for SSL to work, one of the three following assumptions must hold. The smoothness assumption states that points close to each other should have similar labels. The cluster assumption expands on the previous by stating that points form clusters, and points within the same cluster tend to have the same label. Finally, the manifold assumption states that the data lies on a lower-dimensional manifold [3].

Learning algorithms can be split into transductive and inductive methods. In transduction, we try only to predict the labels $y$ for unknown $x$'s. In induction, however, we try to learn the general rule that maps $\mathcal{X}$ to $\mathcal{Y}$ [3]. Label propagation (LP) is one example of a transductive method. In LP, a neighborhood graph of points is created, and unlabeled points inherit their neighbors' label. Label refinement follows the same approach as LP, except all the points have preliminary labels.

### A. Label Refinement Formulation

Weights between neighbors are given as $W_{ij} = exp(-\frac{||f_i - f_j||^2}{2\tau^2})$, in which $f_i$ is the feature vector for point i. For preliminary labels $Y$ and final labels $X$, the objective function is given by:

$$C(X) = \sum_{i,j=1}^{N} W_{ij}\left|\frac{X_i}{\sqrt{D_{ii}}} - \frac{X_j}{\sqrt{D_{ii}}}\right|^2 + \sum_{i=1}^{N}\lambda|X_i - Y_i|^2$$
$$where\ D_{ii} = \sum_{j=1}^{N} W_{ij}. \tag{10}$$

Weights are large if two points have very different features. As the $X$'s are divided by a function of the weight, differences in final labels will not incur a great cost. This makes sense as the difference in feature space suggests that they do in fact belong to different classes. We can also see $\lambda$ controls a regularization term.

### B. Solving the Objective Function

The analytic solution is given by [8]:

$$X = \alpha\left(I - \big(1 - \frac{\lambda}{1+\lambda}\big)S\right)^{-1} Y \tag{11}$$

where $S = D^{-1/2}WD^{-1/2}$, and $\alpha = \frac{\lambda}{1+\lambda}$. In practice, I solved it using both the analytic method (if the number of points was small enough) and SGD. Label refinement at the pixel level would require a $1500^2$ x $1500^2$ matrix. Thus, for larger sets, I did refinement block by block on the neighborhood graph.

## VII. Evaluation

### A. Superpixel and Pixel Level Classification using Superpixel Model

*1) **Grid Search***

The data was split into training, validation, and test sets. The best model was found through grid search by training on the training set and finding the lowest error rate on the validation set. I grid searched over L1, L2, and Elastic Net for regularization method, and $10^{-6}$ to $10^{-1}$ on a log scale for $\lambda$. The grid search results for best parameters for a SVM are found in Table I. LR has only minor differences in the $\lambda$ for Min-Ratio and Majority labeling schemes.

There are a few points of interest. First, Elastic Net was selected for Majority labeling. Elastic Net is a weighted sum of L1 and L2: $\alpha * L1 + (1 - \alpha) * L2$. I used 0.15 for $\alpha$. Geometrically, the contour of Elastic Net lies within the square contour of L1 and the circle contour of L2. Elastic net is usually

compared to L1 in terms of its sparsity enforcement and tends to do well when features are correlated. Unlike L1, which will pick only one out of a group of similar features, Elastic Net will select the whole group [11]. This makes sense for the Majority labeling, as there is a large number of ambiguous superpixels. Because ambiguous superpixels may be labeled as Road, Building, or Other, this means the three classes are likely to have an overlap of similar features. I believe these overlapping features are the groups of correlated features.

Next, I believe that a relatively large value of 0.1 for $\lambda$ is selected for Confidence-75 because there are not enough examples for each of the 9 classes. As such, the model tends to overfit for the clases with few examples, and thus requires greater regularization. Along the same lines, examples in the classes with few examples may be clearly defined by just one or two features. This would explain the L1 regularization method, which enforces sparsity. This could perhaps by alleviated by adjusting $n$ to distribute the examples more evenly.

| Superpixel Training Data Labeling Method | Regularization | $\lambda$ |
|---|---|---|
| Unanimous | L2 | 0.001 |
| Min-Ratio-75 | L2 | 0.0001 |
| Majority | Elastic Net | 0.0001 |
| Confidence-75 | L1 | 0.1 |

TABLE I. GRIDSEARCH RESULTS FOR SUPERPIXEL CLASSIFICATION

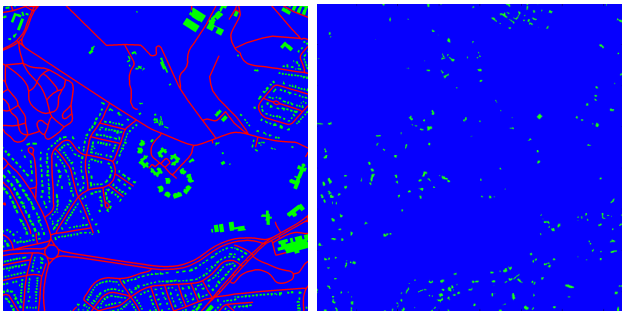
### 2) *Overall Results*

Figure 2 illustrates how inaccurate the final pixel-level predictions are. Pixel level predictions are obtained by simply propagating down the superpixel level predictions. While the distribution of buildings are roughly in the correct area, the buildings are clearly misshapen and sparse. Moreover, all the roads are missing.

Figure 3 plots the accuracy at the superpixel level. We see how accuracy decreases as the uncertainty in the superpixel training label increases. For instance, the accuracy for unanimous is high, as to be expected, because the model is trained and tested only on superpixels in which all its pixels' labels agree. The superpixels are thus well-described. Unfortunately, the conditions for Unanimous and Min-Ratio mean that not all superpixels will be labeled. Furthermore, the vast majority of the unlabeled superpixels contain the Roads and Buildings, so we cannot perform label propagation. As such, we have to turn to either Majority labeling or Confidence. Majority labeling is at the other end of the spectrum of uncertainty in the superpixel training label. It is not surprising that superpixels composed of a wide mix of Road, Building, and Other will be hard to classify. *This points at the reason for the poor final result shown in figure 2. There are too many ambigious superpixels composed of a mix of differently labeled pixels. This is especially true for superpixels around Roads and Buildings, in which the image is noisier and harder to segment.* As a final note, I believe Confidence-75 is especially low because there are not enough examples of each class to capture the feature space.

### 3) *Precision and Recall*

Precision is defined as the accuracy of the predicted positive points. Recall is defined as the percent of positive points that were correctly predicted as positive. As suggested by the result in Figure 2, recall is near 0 for Roads, low for Buildings, and high for Other. Precision is around 20% for Roads (when there are even any predicted Roads), and 70-80% for Buildings and Other. Precision-recall curves are created by varying the decision threshold. As we would like to have both high precision and high recall, a large area under the curve (AUC) is a measure of good classifier performance.



(a) Ground Truth (b) Prediction

Fig. 2. Ground truth and example of poor test results.

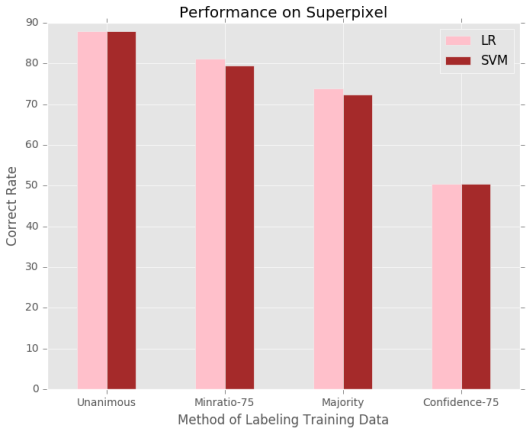



Fig. 3. Accuracy on test set at the superpixel level. Pink = LR, Red = SVM. From left to right on the x-axis, Unanimous labeling (∼94%), Min-Ratio-75 labeling (∼88%), Majority labeling (∼80%), and Confidence-75 labeling (∼58%).

Figure 4 shows the varying precision recall curves for each class when trained and tested using logistic regression with the majority labeling scheme. As we can see, Other has relatively high AUC, while Road has abysmally low AUC. There are two reasons for this. First, Figure 5 shows how few Road points there are. Confusion matrices identify common mistakes in classification by plotting predicted labels against ground truth labels, and we can see that the majority of the points are Other. As mentioned in section 5.C, I tried to guard against this using weighted classes with little luck. Second, as suggested by Figure 3, there are too many ambiguous superpixels. Most of these superpixels contain a large number of Roads and Buildings pixels. These pixels are in inherently noisier regions of the image, and the UCM technique cannot accurately segment them.

### 4) *Feature Weights*

Examination of the feature weights reveals that in general, RGB values were usually among the most important features of the superpixel. The parent size and density were sometimes relevant, and corner density failed to be relevant for Buildings.

Also, as models under Confidence-75 were grid searched to use L1 regularization, the resulting weights are extremely sparse. Under Majority labeling, which uses Elastic Net, only a few features have non-zero weights. Finally, Min-Ratio and Unanimous, which use L2, have smoother distributions across

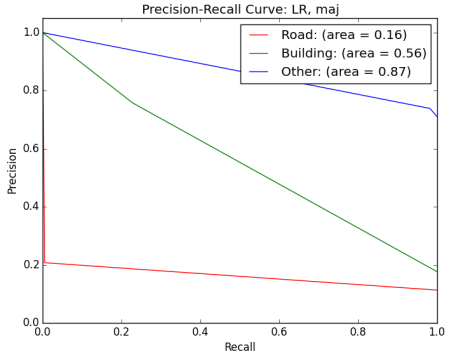



Fig. 4. Precision-Recall Curves for Different Classes: Logistic Regression with Majority Labeling Scheme. Red = Road (AUC = 0.16), Green = Building (AUC = 0.56), Blue = Other (AUC = 0.87).

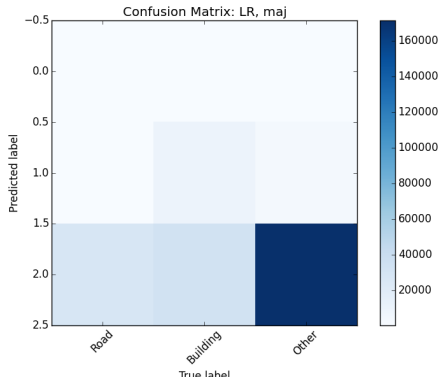
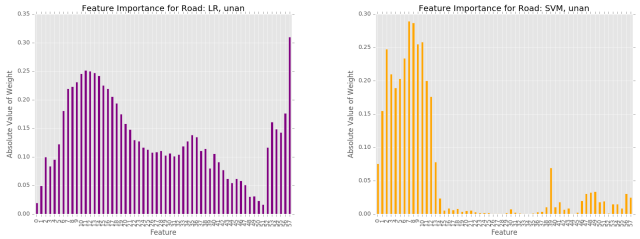



Fig. 5. Confusion Matrix: Logistic Regression with Majority Labeling Scheme. X-axis = Ground Truth. From left to right: Road, Building, Other. Y-axis: Predictions. From top to bottom: Road, Building, Other.

the weights. Relevant figures can be found in the supplementary materials.

Next, in figure 6, we find that fewer features in SVM are selected than in LR. When running SGD on the linear SVM objective, model parameters are only updated if an example violates the margin constraints. Even when L2 regularization is used, this margin update fact can result in sparser models.

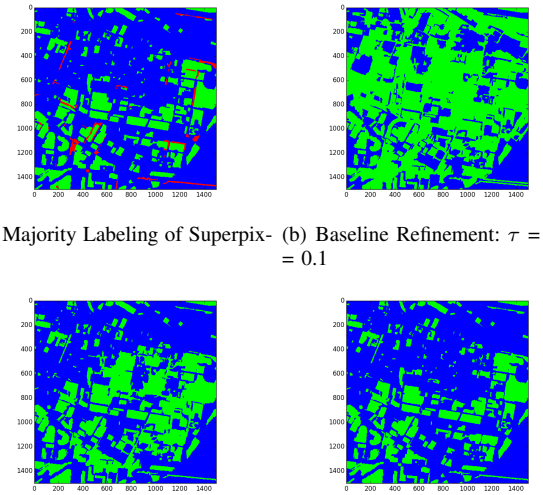



(a) LR (b) SVM

Fig. 6. Feature Importance for Roads with Unanimous Labeling of Superpixels: LR vs. SVM. Y-axis = Magnitude of weight, X-axis = Features. Features from left to right are Binned Intensity (51 features), Corner Density (1), Parent Superpixel Size (1), Parent Superpixel Density (1), average RGB (3).

### B. *Label Refinement using Label Propagation*

Because the preliminary labels from the models are so poor, there is little hope for label refinement. To illustrate that my implementation works, Figure 7 demonstrates label refinement at the superpixel level with k = 0.1 ($\sim$ 1000 superpixels). In image a), superpixels are labeled according to the Majority labeling scheme, and the labels are then propagated down to the pixel level. Image b) shows how the green Building labels are propagated when $\tau = 1$ and $\lambda = 0.1$. Comparing image c) to image b) demonstrates how increasing $\tau$ decreases the amount of propagation. A larger $\tau$ equals a smaller value inside the negative exponent in the weight calculation. This means the weights are smaller, which means that labels are less likely to propagate. Comparing image d) to image b) demonstrates how increasing $\lambda$ decreases the amount of propagation. From the objective function, we can see that a larger $\lambda$ makes it imperative that the difference between X (the assigned labels) and Y (the preliminary labels) is small. Essentially, there is large regularization, and propagation is unlikely.



(a) Majority Labeling of Superpixels (b) Baseline Refinement: $\tau = 1$, $\lambda = 0.1$

(c) Refinement (Effect of $\tau$): $\tau = 1000$, $\lambda = 0.1$ (d) Refinement (Effect of $\lambda$): $\tau = 1$, $\lambda = 0.5$

Fig. 7. Label Refinement on Superpixel Level: Effects of $\lambda$ and $\tau$.

## VII. Conclusion

Ultimately, this pipeline produced poor road and building classification. The superpixels segmentation isn't quite good enough, as there are too many ambiguous superpixels that contain pixels of different classes. Further dividing these superpixels using UCM is not an option - empirical results on further lowering the threshold $k$ below 0.001 did not generate more superpixels. There is a natural limit at which UCM will not further subdivide. The success obtained by Sethi et. al was predicated on their end goal being classification of superpixels, with the ground truth also being provided at the superpixel level [8].

There are still several ways to extend this approach to address the ambiguous superpixels. First, we could try further dividing them. Next, we could use different superpixel segmentation techniques. A popular alternative is SLIC, which uses a k-means based technique to create evenly-spaced superpixels across the image [1]. However, it seems unlikely that this can improve upon the state of the art UCM, which captures global information. In addition, we could also try using fuzzy SVMs, in which class membership is probabilistic instead of absolute [6]. Along the same lines, we could try multi-label classification instead of multi-class classification.

Finally, perhaps the evidence shows that this is not a hopeful approach. We may have to return to CNN's and see if we can train them in a semi-supervised fashion.

## REFERENCES


[1] Achanta, Radhakrishna, et al. "SLIC superpixels compared to state-of-the-art superpixel methods." *Pattern Analysis and Machine Intelligence, IEEE Transactions* on 34.11 (2012): 2274-2282.

[2] Arbelaez, Pablo, et al. "Contour detection and hierarchical image segmentation." *Pattern Analysis and Machine Intelligence, IEEE Transactions* on 33.5 (2011): 898-916.

[3] Chapelle, Olivier; Schlkopf, Bernhard; Zien, Alexander (2006). *Semi-supervised learning.* Cambridge, Mass.: MIT Press.

[4] Farabet, Clement, et al. "Learning hierarchical features for scene labeling." *Pattern Analysis and Machine Intelligence, IEEE Transactions* on 35.8 (2013): 1915-1929.

[5] Harris, Chris, and Mike Stephens. "A combined corner and edge detector." *Alvey vision conference.* Vol. 15. 1988.

[6] Lin, Chun-Fu, and Sheng-De Wang. "Fuzzy support vector machines." *Neural Networks, IEEE Transactions* on 13.2 (2002): 464-471.

[7] Mnih, Volodymyr, and Geoffrey E. Hinton. "Learning to detect roads in high-resolution aerial images." *Computer VisionECCV* 2010. Springer Berlin Heidelberg, 2010. 210-223.

[8] Sethi, Manu, et al. "Scalable Machine Learning Approaches for Neighborhood Classification Using Very High Resolution Remote Sensing Imagery." *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 2015.

[9] Platt, John. "Fast training of support vector machines using sequential minimal optimization." *Advances in kernel methodssupport vector learning* 3 (1999).

[10] Zadrozny, Bianca, and Charles Elkan. "Transforming classifier scores into accurate multiclass probability estimates." *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2002.

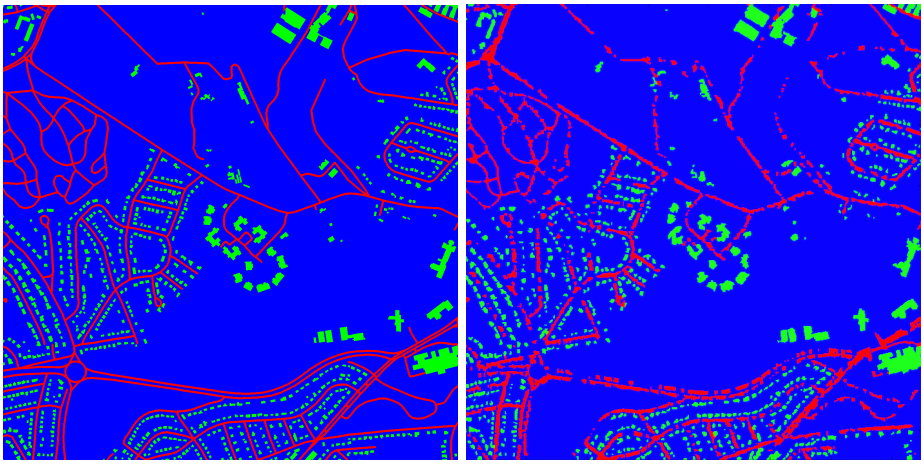
[11] Zou, Hui, and Trevor Hastie. "Regularization and variable selection via the elastic net." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (2005): 301-320.

# Supplemental Materials

## I. Majority Labeling: as good as it gets.

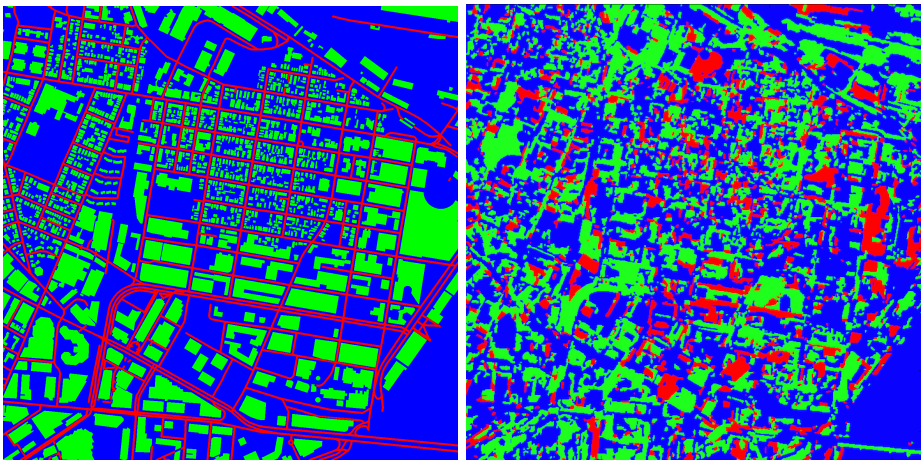
These plots demonstrate what pixel-level results could be achieved if the superpixel classifier achieved 100% accuracy. The right-hand plots are created by assigning each superpixel its Majority labeling.

In figure S1, we see that the best possible result is decent. In figure S2, however, we see just how poor even 100% accuracy can be. In short, the best possible results with Majority labeling are sometimes nowhere close to reality. This is especially true for images with dense roads and buildings.



(a) Ground Truth (b) Best Possible - Majority Labeling

Fig. S1. Ground truth and example of best possible results using Majority labeling scheme for superpixels.



(a) Ground Truth (b) Best Possible - Majority Labeling

Fig. S2. Ground truth and example of best possible results using Majority labeling scheme for superpixels.

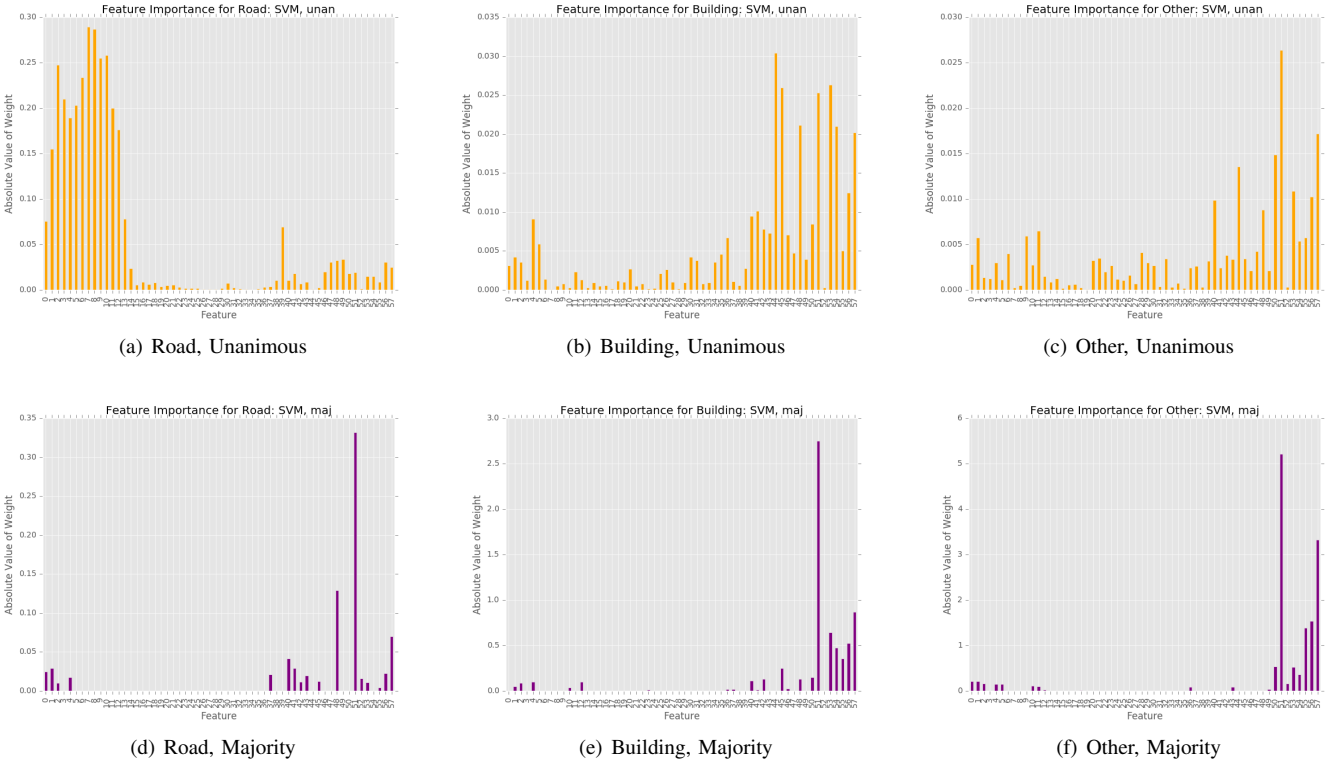
## II. Feature Weights

### A. *Separablility of Roads in Feature Space under Majority vs. Unanimous Labeling*

Figure S3 shows how under Unanimous labeling, Roads can be differentiated from Buildings and Other. Note the importance of the low-end intensity features. This makes sense as roads are a light gray in contrast to the darker blues, browns, and greens found in the rest of the satellite images. Under Majority labeling, however, this separability vanishes. This again reinforces the problem that ambiguous superpixels create.

### B. *Sparsity under different labeling schemes*

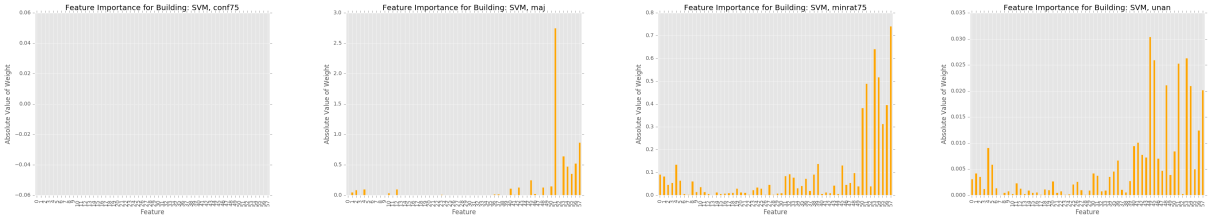
Please refer to figure S4. As models under Confidence-75 were grid searched to use L1 regularization, the resulting weights are extremely sparse. Under Majority, which uses Elastic Net, only a few features have non-zero weights. Finally, Min-Ratio and Unanimous, which use L2, have even more features with non-zero weights.




(a) Road, Unanimous (b) Building, Unanimous (c) Other, Unanimous

(d) Road, Majority (e) Building, Majority (f) Other, Majority

Fig. S3. Feature importance under different labeling schemes




(a) Confidence-75 (b) Majority (c) Min-Ratio-75 (d) Unanimous

Fig. S4. Feature importances and sparsity under different labeling schemes