Project Proposal - MapScan

Rafael Ferreira (9081805708), Saym Imtiaz (9082427411)

September 2020

1 Problem Motivation

Maps currently available on the Internet like Google Maps, Bing Maps or Mapbox do not have a lot of detail about the terrain or land usage (farms, forests, type of vegetation, biomes, etc). They usually separate the areas in either urban (gray), rural/forests (green), or water (blue), as demonstrated in Figure 1.

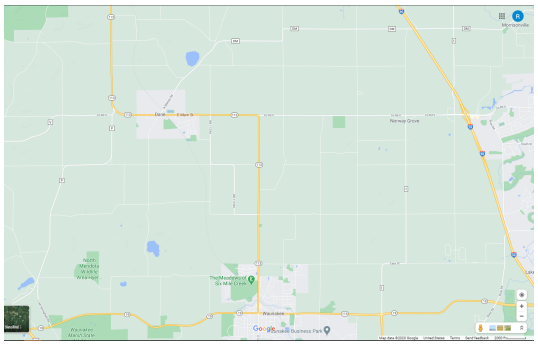


Figure 1: Map View on Google Maps of Waunakee and Dane Wisconsin

The same area can be seen in much more detail in Figure 2. This image presents the satellite view from Google Maps.

Figure 2: Satellite View on Google Maps of Waunakee and Dane Wisconsin

Here it is possible to see some of the green labeled areas are farms, some are forests, and the borders between cities and nature are actually much smoother than shown in the map view, where they are usually shown as squares.

Our proposal, known as MapScan, is to evaluate the power of computer vision algorithms to segment and classify regions of the planet in much more detail than currently available, making a distinction between farms and forests, different biomes (desert, rain forest, temperate forest, savannas, etc), and more accurate city borders, for example. This increased segmentation of satellite imagery can then be used for a multitude of tasks, such as improved visualizations of the world and quantifying land usage automatically. Such automatic land usage computations have the potential to be incredibly useful in evaluating land usage and urban sprawl and therefore make informed urban planning decisions.

2 Existing Work

There currently exists a great deal of literature in the field of satellite image processing. Satellite image processing has been used to for specific tasks such as monitoring deforestation rates [1], classifying crop types for crop yield estimation [2], or determining areas prone to flooding [3].

In these works, a number of different techniques have been proposed to solve domain-specific problems. Such techniques include and are not limited to Transfer Learning, Autoencoders, Convolution Support Vector Machines (CSVM), and Convultional Neural Networks (CNNs). Much of the current research in this domain involves data collection of detailed satellite imagery, feature extraction, and training to create models that can infer information about satellite imagery. However, there is still much exploration in this field to which methods and techniques are best equipped to process and analyze immense amounts of satellite imagery.

One work we found to be particularly similar to our proposal is NASA's Land Cover Type product MCD12Q1v006 [4]. However, it aims to classify land cover of the whole planet, so the resolution is not that high (500 meters squared per pixel), and whole areas are classified as one single type. Conversely, our

objective is to scan only a portion of the state of Wisconsin, but in much more detail.

3 Approach

The general approach is to collect satellite imagery, label images for training and validation, and then evaluate the accuracy of different algorithms.

3.1 Image Collection

A few changes were made to the initial proposal. First, we decided to only use satellite images from Mapbox API. We extracted 224x224 resolution images from the area between coordinates 42°17'22.92" N and 89°37'52.68" W to 44°21'5.4" N and 87°49'51.24" W. This region corresponds to from Freeport, IL to Green Bay, WI, and notably includes both Madison and Milwaukee. For the purposes of training and validation in this report, a zoom level of 13.5 was used.

A python script was also used to collect a database of images for future algorithms. This script utilized the Mapbox API and collected every single image within the aforementioned region at a zoom level of 14. This database contains about 10,000 images, and are tiled sequentially in a manner that makes it considerably easy to stitch together in multiple sizes and configurations. This database will be especially helpful for using learning algorithms such as U-Net and DeepLab.

3.2 Segmentation Classes

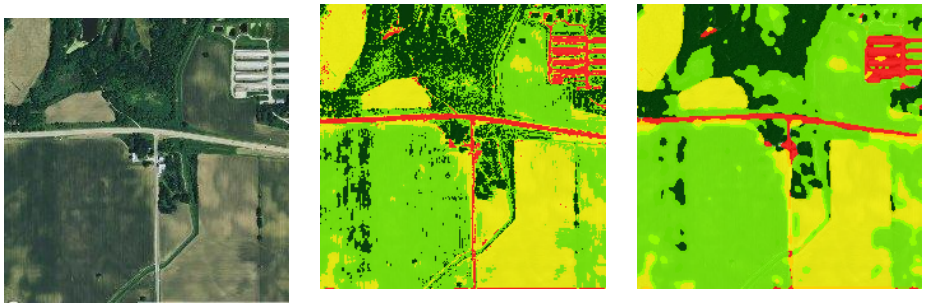
We used 8 different classes for the segmentation:

- Urban areas, covering roads, buildings and residential areas
- Forest, covering areas densely populated by trees
- Croplands 1, covering crop/livestock farms with a more yellow coloring
- Croplands 2, covering crop/livestock farms with a greener look than Croplands 1
- Croplands 3, covering crop/livestock farms with a strictly green coloring
- River, covering bodies of water shaped like a river, with narrow paths
- Lake, covering bodies of water shaped like lakes and ponds
- Grass, covering open green fields that do not belong to farms

The idea behind separating "Grass" from "Croplands" is that we could later calculate the total proportion of farm lands in a region.

3.3 Image Labeling

The first challenge we faced was to label the images. It was too time-consuming to label every region of every image completely manually, so we developed a tool to help us. First, we used the k-means clustering algorithm on each image, choosing the number of clusters based on the number of classes we expected to be present in it. Since the cluster ids resulting from k-means are random, we had to link each cluster found to one of our defined classes. We then found out there was a lot of noise in the resulting labels, so we applied a Gaussian filter on each class channel separately, and each pixel was set to the class with the highest corresponding value. The results of this process are shown in Figure 3.



(a) Satellite image

(b) K-means result using 4 clusters

(c) Labelled image after applying Gaussian filter

Figure 3: Example of the labelling process

The developed tool allowed for downloading a random image, setting the desired number of clusters, and then defining the links between cluster ids and classes. It then outputs two text files containing the labels before and after applying the Gaussian filter, so that results could be compared later.

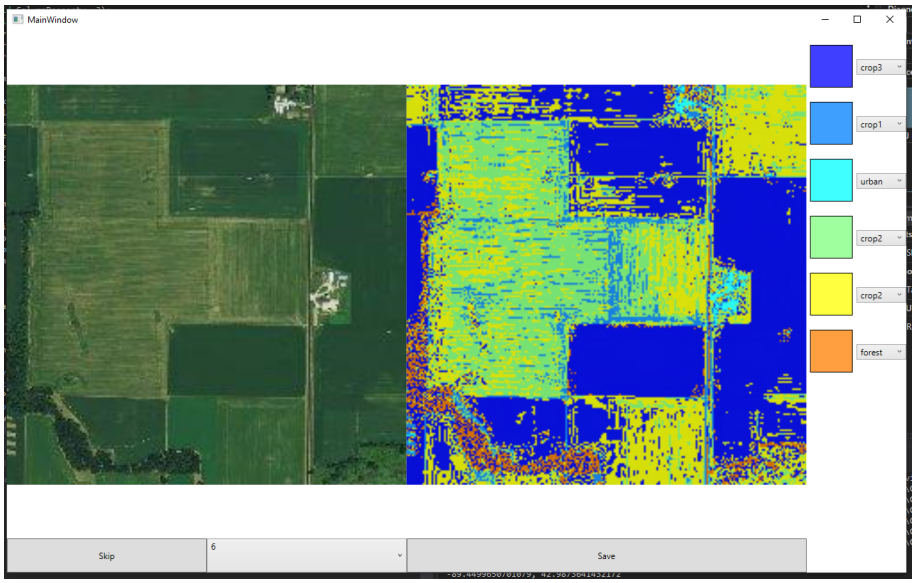
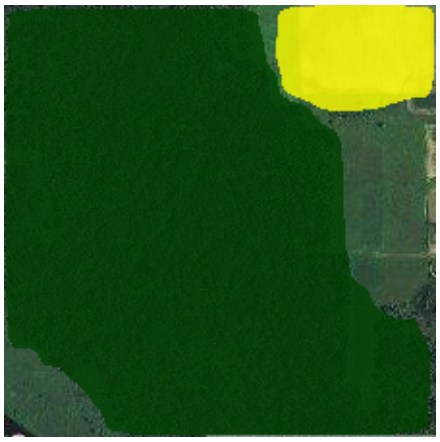


Figure 4: Tool used for labelling the images

3.4 Algorithms

The next step in the process of implementing MapScan was to evaluate a number of different algorithms. One such algorithm was Mask R-CNN, which predicts an object mask given an input image. After implementing Mask R-CNN, an informal qualitative assessment of the algorithm indicated that it was not suitable and effective for the goal of the project. Instead of finding a segmentation mask for the whole image (i.e. setting every pixel to a corresponding class), the algorithm is more suitable for object detection and instance segmentation. In other words, it finds candidate bounding boxes in the image corresponding to objects present in the training set (for example, a patch of trees), and then applies a mask to each object found, based on its expected shape. Because of that, it does not guarantee that every pixel will be included in an object. This does not fit the intended goal of labeling every pixel in an input image. Figure 5 shows an example of an output from Mask R-CNN after training with 30 images, where we can see it does not fill the whole image.

(a) Satellite image



(b) Output overlay

Figure 5: Example of Mask R-CNN output

Because of that, we decided to exclude Mask R-CNN from the list of algorithms being evaluated, and we replaced SegNet for U-Net, as it achieves better results in other segmentation tasks, according to the literature. Finally, we included a baseline to compare with the deep learning-based methods. The baseline method consisted of calculating the mean pixel values for each class, and then setting each pixel of the new image to the closest class, considering the Euclidean distance across Red, Green and Blue channels.

$$d(p, c) = \sqrt{(r_p - r_c)^2 + (g_p - g_c)^2 + (b_p - b_c)^2} \tag{1}$$

One problem we found with the baseline method is that some classes have very similar mean pixel values, as the difference between them depends exclusively on the context. For example, we can see in Table 1 that Croplands 3 and Grass have very similar mean pixel values.

classId	className	(R,G,B)
0	urban	(170, 181, 160)
1	forest	(27, 50, 37)
2	crop1	(129, 137, 106)
3	crop2	(73, 100, 73)
4	crop3	(51, 86, 59)
5	river	(6, 30, 16)
6	lake	(36, 38, 38)
7	grass	(51, 82, 61)

Table 1: Mean pixel color for each class

Thus, the final list of algorithms to evaluate are:

- Mean pixel value (baseline)
- U-Net
- DeepLab (DeepLabv3+)

4 Results

One critical challenge in providing the results of the algorithms used is what provides the best measure of correctness. There are many different ways to measure correctness, some more suitable than others in the field of image segmentation. Furthermore, even with image segmentation, the ultimate goal of the use case influences which measure should be used. To evaluate the effectiveness of each measure, 8 random images were labeled and then compared to the predictions made by the mean-pixel algorithm described in Section 3.4.

The first measure of correctness used was to count how many pixels were correctly labeled by the mean-pixel algorithm. The accuracy is then given by Equation 2. In applying this metric on the 8 images, the mean-pixel algorithm had an average accuracy of 49.7%. However, the standard deviation of accuracies between the 8 images was 16.5%, which is quite high. This metric was thus considered not adequate since it does not provide any indication of how accurately some labels were applied compared to others nor does it make any attempt of considering false positives and false negatives.

$$Accuracy = (TotalRight)/(TotalNumberofPixels) \tag{2}$$

Another measure of correctness considered was the Hausdorff distance. This metric is a measure of how far two subsets are from each other. Hausdorff distance is commonly used in computer vision to find a template in a given target image. The motivation in using it here is to find how far off the mean-pixel algorithm mask is from the true labels. The average Hausdorff distance was given as 40.0, and the standard deviation between each distance was 7.2. One advantage Hausdorff distance has is that there is significantly less standard deviation between each image pair for the same algorithm used. However, Hausdorff distance is hard to convert to a notion of percent accuracy, and it is sensitive to the numerical values of labels. For example, a pixel labeled as 8 that was supposed to be labeled as 1 will produce a greater Hausdorff distance than had that pixel been labeled 4. Thus, Hausdorff distance was ultimately decided not to be used as the metric for comparing the correctness of algorithms.

The last measure of correctness considered was the Sørensen–Dice coefficient. This measure is given by Equation 3. The Sørensen–Dice coefficient is a measure of similarity between two samples like the Hausdorff distance. However, it only provides values between 0-1 similar to percent accuracy, in which perfectly identical samples have a coefficient of 1. The coefficient also has the advantage of rating the correctness for each label. For example, with our use case of 8 different labels, computing for the dice coefficient produces an 8 element vector.

This vector can also be averaged to get an overall accuracy of the algorithm in question. Since the Sørensen–Dice coefficient has the advantages of being easy to compute, provides the accuracy of each label as well as an overall accuracy measure, it was selected as the measure of correctness for this project.

$$DSC = \frac{2TP}{2TP + FP + FN} \tag{3}$$

where:

TP = True positive

FP = False positive

FN = False negative

The overall accuracy of the mean-pixel algorithm using the Sørensen–Dice coefficient was 65%, with a standard deviation of 13.8%. Table 2 gives the average coefficients for each labels. Among the 8 images analyzed, there was no lakes and rivers, and thus are not present in the table. From the table, it is clear to see that the mean-pixel algorithm works relatively well with classifying urban, croplands 1, and croplands 2 regions. However, as previously discussed, croplands 3 and grass regions share very similar mean pixel values. This is reflected as both labels have quite low average coefficients. From these results, it is clear that mean-pixel only works well when each region contains considerably different pixel values, which is not true for our use case. Thus, it is necessary in future work to improve upon this baseline.

Label	**Average Coefficient**
Urban	0.870
Forest	0.620
Croplands 1	0.755
Croplands 2	0.822
Croplands 3	0.518
Grass	0.272

Table 2: Sørensen–Dice Average Coefficient for each Label

5 Future Work

Listed in Table 3 is the expected timeline for the project implementation. Major action items remaining include implementing U-Net and DeepLab.

In general, in looking at other algorithms, a future segmentation accuracy of 85% is targeted. This is on par with similar work done with learning algorithms in the field of satellite image segmentation. After evaluating different algorithms such as U-net and Deeplab, it would be exciting to apply the selected algorithm to perform image segmentation tasks. For example, we are interested on applying our future algorithm on much large patches of satellite imagery such as all of Greater Milwaukee. From there, it then becomes possible to calculate

Task	Completed By
Explore satellite image sources	October 5th
Extract images from chosen source	October 12th
Label Images (segmentation and classes)	October 19th
Implement K-mean clustering	October 23rd
Implement Mask R-CNN	October 26th
Midterm Report	November 2nd
Implement U-Net	November 16th
Implement DeepLab	November 23rd
Final Results, Discussion, and Report	November 30th

Table 3: Project Timetable for MapScan

land usage and proportions, such as how much of southern Wisconsin is used for agriculture versus urban sprawl. Such quantities can be quite useful for smaller municipalities as a tool for them to use to make informed decisions on urban planning or land allocation.

Thus, the overall goal is to empirically evaluate different image segmentation algorithms, and then develop MapScanner to be a useful tool for automatic land allocation calculations to an acceptable fidelity.

6 References

1. http://www.lvc.ele.puc-rio.br/projects/ChangeDetection/index.html
2. http://www.lvc.ele.puc-rio.br/projects/CRF_CropRecognition/home.html
3. https://www.sciencedirect.com/science/article/pii/S1464343X19301529?via%3Dihub
4. https://lpdaac.usgs.gov/products/mcd12q1v006/