Milestone: Extracting Map Features from Satellite Images

Igor Berman
EE PhD, Stanford
igorber@stanford.edu

Abstract

In this project we propose a way to extract useful map features from satellite images using neural networks. We first train a classifier network to do single-pixel labeling and then transfer the learned weights into a fully convolutional network that outputs a segmented map of the input image. In addition, we show that by adding the class Building we improve performance on other, similar looking inputs.

1. Introduction

Satellite images are the main source of information for modern maps, and yet they still need to be manually augmented and corrected to extract desired features like roads and buildings. One reason for human involvement is the inherent "noise" – different image quality as well as light conditions, coloring and occlusion. The other is ambiguity – useful map features are often abstract and hard to define in image-processing terms. Even roads can differ significantly - an interstate highway has very little in common with a local driveway in a suburb.

In this project we aim to automatically extract and label useful features from raw satellite images, akin to transforming "Google Earth" images into "Google Maps" maps. We treat this as a segmentation problem, assuming that each pixel on the satellite image belongs to one of four possible classes: Road, Water, Building and Background, where Background is a catch-all default class. In order to design a segmentation network, we first train a classifier network that only classifies a single pixel into one of the above labels.

1.1. Related Work

Methods of using neural networks to extract geographic features were shown in [1][2], though these works focused exclusively on locating roads and did not take advantage of classifying networks. Conversely, in [4] pre-trained classifying networks were modified for image segmentation, associating parts of the image with ImageNet classes. This work also encountered the problem of resolution reduction due to subsampling and described several solutions.

2. Data

Data was obtained using the Google Maps API, which provides acces to satellite images as well as color-coded maps. Since the standard map format is overly-rich in color and features, several "Style" flags were invoked to present only the four required classes in clear color (Figure 1). The images are all from the greater San Francisco area, and are comprised mostly of urban localities.

The original Sattelite images, of size 640x640, are divided into 64x64 patches, each corresponding to a map area of roughly 20m x 20m. Each patch has a label, which is just the correct class of its central pixel. The correct class is determind according the color of the corresponding pixel in the map image of the same area. Each 64 x 64 pixel patch was fed as trainig data to the classification network. Overall, 200K images are used as trining set, and 20K are divided equally between the validaiton and test sets. We made sure that each class appeared with roughly the same probability in the training data.

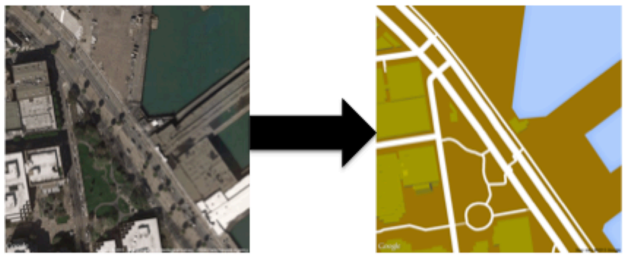

Figure 1: A Satellite image and the corresponding map, as obtained from the Google API. Each of the 4 classes is represented with a different color.

3. Technical Approach

Our technical approach is similar to [4]: First, we train the Classification Network. This is a standard network that takes in a 64 x 64 patch of satellite image and trains to classify its central pixel. Then, we transfer the weights to the Segmentation Network: a fully convolutional network that acts on a larger satellite image and produces a segmentation map. Finally, we apply some post-processing to create a full size 640 x 640 segmentation map.

Note that our initial approach, described in the Project Milestone, treated this as a denoising problem, similarly to [3]. The network was trained to map each pixel from RGB to "correct" RGB, where the unmodified Google Maps map was treated as the ground truth and the satellite image as the noisy version. The RGB-to-RGB transformation that the network learned tended heavily towards the mean of the image, and could not extract useful map features. This approach was not pursued further and will not be discussed in this report.

3.1. Classification network

Our final Classification network is comprised of 5 Convolutional layers and 2 Fully Connected layers. Dropout with probability 0.5 is applied to the last Conv layer and the first FC layer. There are no pooling layers in the network, as these complicate fine-tuning the final segmentation network. This limits the maximum depth of the network and slows training, but allows for higher resolution in the resulting Segmentation map. It also limits the receptive field of each layer, but testing has show that this doesn't affect performance.

3.2. Segmentation Network

Having trained the Classification Network on 64 x 64 patches, we now transfer the learned weights to the segmentation network in the following manner (Figure 2) :

a. Convolutional layers remain unchanged.

b. Fully Connected layers are replaced with Convolutional layers. The first new Conv layer has a receptive field of the same size as that of the analogous FC layer. The next layer has a receptive field of spatial size 1. Depth of the layers is unchanged.

The Segmentation network takes satellite image patches of size 160 x 160 and produces a 160 x 160 x 4 probability map, where each pixel is assigned a softmax "probability" to belong to each of the 4 classes.

3.3. Segmentation and Post Processing

The entire image segmentation pipeline is as follows: 640 x 640 satellite image is divided into overlapping patched of 160 x 160. Each patch is fed through the Segmentation Network, producing a per-pixel softmax mapping. The overlapping segmentations are accumulated back into a 640 x 640 x 4 probability mapping, and the argmax class of each pixel is taken as the prediction. The resulting image is passed through a median filter of size 9 x 9 to remove some noise and irregularities.

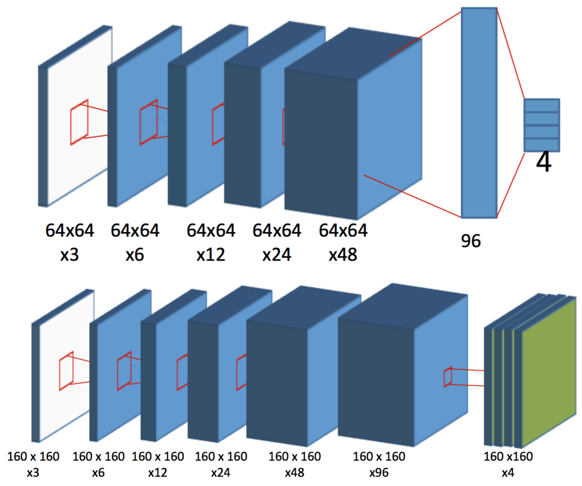

Figure 2: The Classification Network (Top) and the Segmentation Network (Bottom). Note that the two fully connected layers have been replaced with convolution layers.

4. Experiment

4.1. Training the Classification Network

In our initial effort to construct an efficient classifying network, we first used only three classes: Background, Road, and Water. Buildings were labeled as Background. This led to a relatively high error rate of 26%, as can be seen in Table 1 (Error rate is calculated as unweighted average over all classes, meaning – the error rate if pixels of all classes appeared with the same frequency. If we were to use the weighted average, our classifier would be heavily biased towards Background).

Prediction / True Label	Background	Road	Water
Background	70%	28%	2%
Road	32%	65%	3%
Water	5%	2%	93%

Table 1: Performance of a 3-class classifier

The most obvious source of error in this classifier is the confusion between Road and Background. This can be attributed in large part to buildings – flat gray surfaces that resemble roads but are labeled as background (Figure 3).

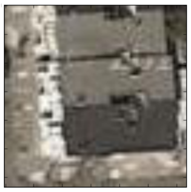

Figure 3: roof (Background) classified as Road

To tackle this problem, we added Building as a possible label, separate from Background. After training the classification network for ~40K iterations (of 50 batches each), we achieved an error of 20%. Improving classification of both Road and Water, albeit slightly degrading the classification of Background.

Prediction True Label	Background	Road	Water	Building
Background	63%	18%	1%	18%
Road	12%	78%	0%	10%
Water	1%	0.5%	98%	0.5%
Building	8%	11%	0%	81%

Table 2: Confusion matrix for final Classification Net

Further testing has been done with larger filter sizes in the Conv layers, but it did not improve performance.

4.2. Transferring Weights to Segmentation Network

Our next step was to transfer the weights to a segmentation network, as outlined in 3.2. Having run it on full-sized satellite images and applied a median filter, we obtain the results shown in Figure 4:

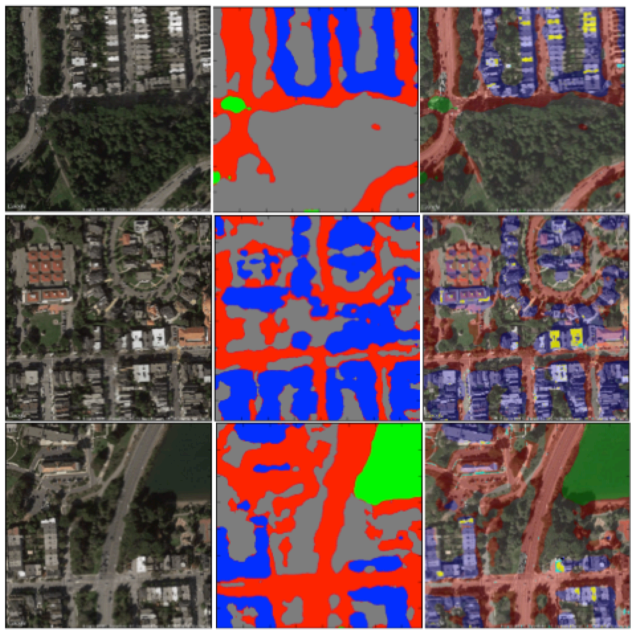
Figure 4: Left: Original satellite images. Center: Segmentation results showing the 4 classes: Background (gray), Road (red), Water (green) and Building (blue) / Right: Segmentation overlaid on image.

We see that most useful features are represented in the segmented image, although some are deformed. It's also worth noting that occluded or shaded areas are still correctly classified: In the first image, a horizontal road is mostly hidden by trees and shades, but still present in the output. In the third image, a tree casts a dark shadow over the lake, but the area in the shadow is still correctly classified.
Still, the post-processing leaves a lot to be desired. The median filter closes small gaps and smoothens edges, but it does not "encourage" straight lines or continuous shapes, and it distorts objects that are naturally angular, like buildings.

4.3. Alternative approach

In addition to the tests described above, we also tried to train a classification network by fine-tuning AlexNet. We used the first 5 layers of Alexnet (4 Conv, one FC), and added a final FC layers that maps to the 4 classes.
This fine-tuned net achieved better single-pixel classification performance (15% error), but could not be easily transformed into a segmentation network. The reason is that each pooling layer subsamples its input, which makes the output have a much lower resolution than the input. Assuming 2-pooling, the output would have only 4^{-N} as many pixels as the input. A few solutions were explored in [4], with the simplest one being the Shift-and-Stitch, in which up to 4^{N} copies of the input are produces using shift-by-one, and their results are then interleaved.
In our case, this translates to running each 640 x 640 input through the network 256 Times, with proved impractical given our time and memory constraints. Instead, we attempt to use shift-by-4 and run in only 16 times through the network, using bilinear interpolation to stretch the results into the desired resolution. However, this gives unsatisfactory results, as can be seen in Figure 5:

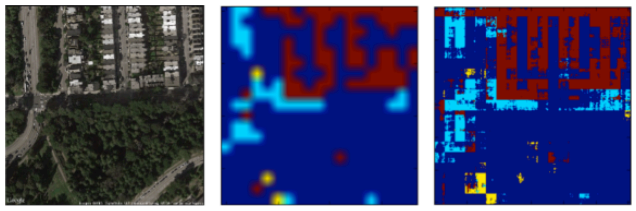
Figure 5: Left: original image. Center: output of Segmentation net naively constructed from Alexnet-based Classification net. Right: a partial Segmentation net that uses bilinear interpolation.

5. Conclusion

In this project we've demonstrated that Convolutional Networks achieve high performance on classifying geographic images. By adding the class Building, we've actually improved performance on all other classes, by making the default Background class better defined.
We then used the weights we learned in the classification task for satellite image segmentation, without requiring any further training, and with very basic post-processing on the output. This was made possible by the fact that we didn't use pooling after the Conv layers.
However, one weakness of this approach is that given the outputs of the one-before-last layer, all pixel class predictions are independent. In other words, we fail to incorporate our knowledge about typical pixel distribution, like the fact that roads are generally elongated and maintain a constant width. In [4] this is solved by training another neural network. We only use the basic Median filter.

References

[1] Jain, Viren, and Sebastian Seung. "Natural image denoising with convolutional networks." Advances in Neural Information Processing Systems. 2009

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

[3] Mnih, Volodymyr, and Geoffrey E. Hinton. "Learning to label aerial images from noisy data." Proceedings of the 29th International Conference on Machine Learning (ICML-12). 2012.

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