Group 97: Road Segmentation

Laurent Lejeune, Tatiana Fountoukidou, Guillaume de Montauzon

I. Introduction

The goal of this project is to classify roads in satellite images. An example is shown in figure 1. This report is structured as follows. We start from a baseline where basic features are extracted and classified. We then introduce a more elaborate attempt using a feature set made of Scale Invariant Feature Transform (SIFT) and Hough line transform among others. Generic classifiers similar to our baselines are then applied and their predictions are refined using a structured output model called structured SVM. A third attempt, by far the most effective, makes use of a Convolutional Neural Network. The next sections describe in more detail the steps that were followed, and their results.

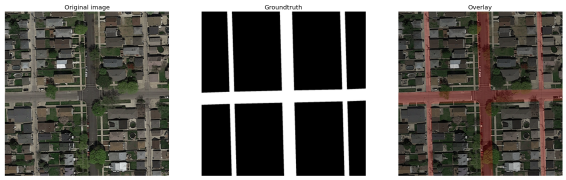


Figure 1: Example of training image with its ground-truth

II. Data exploration

The provided training set contains 100 images of size 400x400 along with their ground-truth. A total of 6 images are discarded because they either show too few positive class pixels, or some misleading regions such as rail-tracks. We notice that most images are made of grid-like roads, sometimes occluded by trees.

III. Baselines

A. Comparison of classifiers

The images are first cropped into non-overlapping square patches of size $d = 16$. As features, the mean and variance RGB values are extracted for each patch. As for the generation of ground-truth samples, a patch $p_i \in \{0;1\}^{d \times d}$ is considered a positive sample if $\frac{\sum_j p_i}{d^2} > 0.25$, and negative otherwise. A grid search was performed on the hyper-parameters of three different classifiers: Logistic regression, SVM with RBF kernel, and random forest. To evaluate the performance, a 5-fold cross validation is run over the training dataset, and the F-score on the testing set is calculated for each fold. The mean and the variance of the F-score over the 5 folds is then calculated. The scores for the optimal values of hyper-parameters are shown in table **??**. Let us note that random forest is significantly faster to train than SVM. This drives our choice to discard the latter for the remaining of this work.

Classifier	$F - score$(**mean** $\pm$ **std**)
Logistic regression ($\lambda = 1e-2$)	0.44 ± 0.02
SVM (rbf kernel, $\lambda = 1e5$)	0.5 ± 0.02
RF (50 trees, max depth = 10)	$\mathbf{0.5 \pm 0.02}$

Table I: Baselines

B. Influence of patch size

We examined the impact of the patch size on the classification performance. The mean and the variance of the pixels was again used as a feature, and the classifier was the baseline RF described above. The F-scores for the different patch sizes are shown in table II. Even though the larger the patch size, the better the F-score, we observed that many roads were too narrow to be represented by a larger patch size. The higher score in this case is misleading, since it is calculated on a patch level, and not on a pixel level. Larger patches are more likely to have more than 25% road pixels, but this does not mean that they fully represent a simple road block. This way, although more patches are classified correctly, the overall segmentation mask that comes up is not representative enough. For this reason, the patch size of 16 was kept, as more appropriate to capture the different road widths. For the rest of the work, the performance of the random forest (RF) classifier for patches of size 16 will be used as the baseline.

Patch Size	8	16	25	32
F-score	0.45 ± 0.03	0.5 ± 0.03	0.54 ± 0.04	0.55 ± 0.06

Table II: Patch size comparison

IV. Structured SVM approach

Prior to feature extraction, our images are pre-segmented using SLIC Superpixels (Simple Linear Iterative Clustering) [1], which groups pixels in mid-level regions in an iterative manner. The algorithm starts

from a regular grid of cluster centers and iteratively updates the labels of their neighboring centers based on a distance measure. Compared to square patches, superpixels allow to extract more discriminative regions. The next section describes the feature extraction and their encoding.

A. Feature extraction

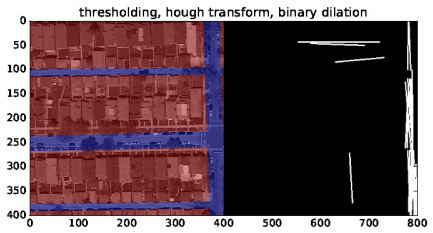


Figure 2: Example of a hough transform. Left: Input image with the ground-truth overlay. Right: 20 lines with lowest color variance.

Following an exploration of the related literature [2], we select a set of features to extract.

- *SIFT (Scale-Invariant Feature Transform) [3]*: This descriptor is used extensively in computer-vision applications. It computes a histogram of oriented gradients on 16×16 windows centered at a keypoint and gives a descriptor of 128 scalar values. The keypoint detection step is not performed, instead we extract the descriptors on a dense grid at canonical scale and orientation. As advised in [1] to improve illumination invariance, the integer value descriptors are first normalized to unit-norm, ceiled to 0.2, and renormalized to unit-norm. As we require that each segment be represented by a single feature vector, we encode the dense SIFT descriptors contained in a given segment in a "bag-of-features" manner through the following steps:
  1) Based on a sufficiently large number of SIFT descriptors computed on 10 images, we start by fitting a PCA model. We have checked that the explained variance at 60 components is above 99%.
  2) A codebook is generated on the aforementioned training samples. A codebook is merely a set of K-means clusters that is used to encode the input (compressed) descriptors to integer values.
  3) We then compute a normalized histogram of codes (bag-of-features) in each segment. This gives us a single texture feature vector for mid-level regions.

[1] https://people.csail.mit.edu/hasinoff/320/sift-notes.txt

- *Hough line transform.* This transformation has already been used in a state-of-the-art method [4]. First, the edge map is computed using a canny edge detector[5]. Given some parameters, a set of lines are extracted on the edge maps and sorted based on their RGB variance, i.e. we want to keep the lines along which the color variations is minimal. An example is shown on figure 2. As feature, we take the mean value of the hough map on the segment.
- *Euclidean distance transform.* This straightforward transform is used to compute, at each pixel location, the shortest "taxicab"[6] distance to an edge pixel. Using as input the canny edge map, we expect this feature to discriminate cluttered regions such as those containing blocks of buildings where the Euclidean distance tends to be smaller. The mean value over the segment is used.
- *Mean RGB value.* Roads tend to have greyish colors.

To summarize, our feature extraction procedure provides a total of 65 features per superpixel.

B. Refinement of generic models using structured SVM

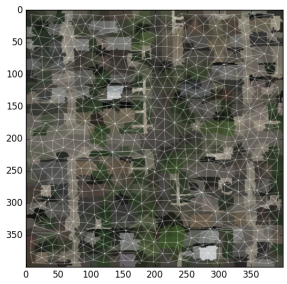


Figure 3: Example of a superpixel-segmented image with connecting edges.

Using structured models [7], one can leverage the spatial relations between mid-level regions. As shown on figure 3, a segment considered as road gives a strong prior on the "roadness" of its neighboring segment. This is formalized as an undirected graph on which the node features are assigned unary potentials. In our case, the unary potentials are given by probability estimates given by any generic models such as logistic regression or random forest. Inspired by [2], the edge costs are made off of two features: The difference in mean LUV color, and the number of pixels that separate two segments (length of separating path). This last feature allows to penalize segments that are "weakly" connected. Indeed, we have verified visually that roads tend to be composed of regular chains of square-like segments, thereby justifying that choice.

Formally, structured models aim at maximizing an energy functions of the form:

$$E_w(X,Y) = \sum_{i \in \mathcal{V}} E_{data}(y_i; x_i) + \sum_{i,j \in \mathcal{E}} E_{smooth}(y_i; y_j) \\ = \mathbf{w}^T \psi(X,Y) \tag{1}$$

Where $\mathcal{V}$ is the set of vertices representing a segment and $\mathcal{E}$ are the edges. The data and smoothness term are combined in the joint-features vector ψ. The variable y represents the structured labels. The groundtruth labels are extracted analogously to the patch case, meaning a superpixel is labeled as road if more than 25% of its pixels belong to a road. In our setup, any probabilistic regression model (logistic regression, random forest, e.t.c.) can be used for the data term. Following [2], the pair-wise edges potentials are given by:

$$\phi(c_i, c_j | s_i, s_j) = \frac{L(s_i, s_j)}{1 + \|s_i - s_j\|} \tag{2}$$

Where c and s are the mean LUV-space colors. The function L expresses the length of the shared boundaries between two segments. The provided code relies on the pyStruct package [8], which implements the cutting-plane algorithm proposed by Tsochantaridis et al [7].

V. Classification

Two methods were implemented. The first consists of feature extraction and classification, with the features mentioned aboved. The second consists of training a convolutional neural network (CNN) that takes as input the image and outputs a segmentation mask. Both approaches are described below.

A. Classification based on feature vectors

The features described above were used to train a random forest ensemble classifier. A grid search to define the hyperparameters of the classifier was performed, and a forest of 50 trees with maximum depth of 10 was chosen. The CRF refinement step was then performed, and further improved the classification result.

B. U-net

A convolutional neural network designed for image segmentation was used, as described in [9]. U-net is a fully convolutional neural network, meaning that it consists exclusively on convolutional and pooling layers, and not fully connected layers in the end. As a result, the output of the network is not a class assignment for the input, but a feature map. In our case the output of the network is a segmentation mask for the input image. The structure of the network is shown in figure 4 [2]. In the first part of the network, the input image is filtered multiple times, and max pooling is performed to reduce the dimensions of the feature maps. In the second part, the representations that are learned in all the different network depths are combined and up-sampled again, so that an image mask of size equal to the imput's can be constructed.

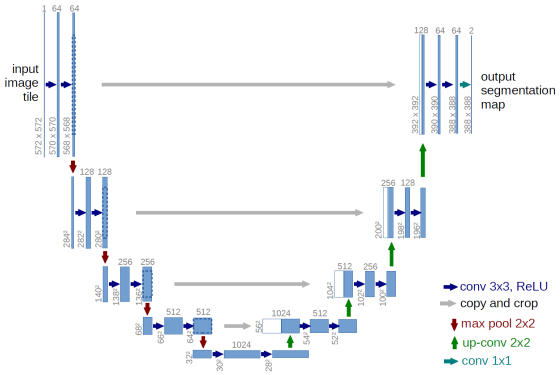
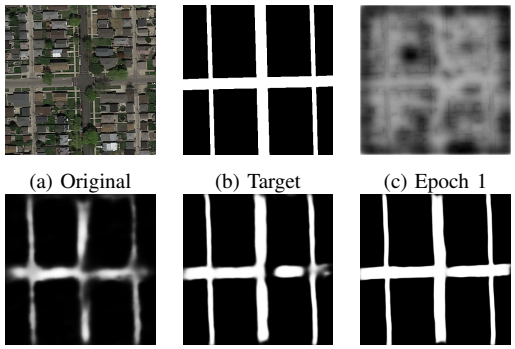


Figure 4: U-net [9]

1) Hyperparameter selection: The network is trained for 100 epochs, but the instance with the lowest validation loss is kept. The loss function is categorical crossentropy, and an Adam optimizer with learning rate of $l = 0.0001$ is used.

2) Data augmentation: In order to train such a network (23 convolutional layers), data augmentation is necessary. After dividing our set to training and validation, we therefore augmented the training set, by randomly rotating, flipping, and rescaling the images. An example of some consecutive steps of the network output during training is demonstrated in fugure 5.

The experiments for the U-net were run on the keras[10] framework.



(a) Original (b) Target (c) Epoch 1

(d) Epoch 5 (e) Epoch 30 (f) Epoch 100

Figure 5: Example Unet output during training. Note that the image is not part of the training set.

[2] In our case the input size is $400 \times 400 \times 3$, and the output is of size 400×400

VI. Results

The performance with respect to the regularization parameter for the feature classification case is demonstrated in figure 6 The results of a 5-fold cross valida-

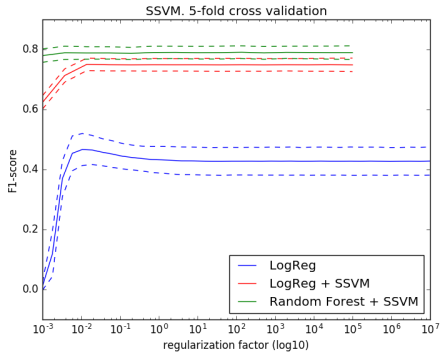


Figure 6: Feature classification performance for different regularization parameters.

tion, for the different classification approaches that were attempted and described above are shown in table III [3]. Also, in figure 7 one can see an example image with

Classification method	$F-score$(**mean** $\pm$ **std**)
Random forest, square patches, basic features [Baseline]	0.5 ± 0.02
Logistic regression ($\lambda = 1e-2$), elaborate features	0.45 ± 0.02
Random forest (50 trees, max. depth 10), elaborate features	0.68 ± 0.03
Random forest, SSVM refinement, elaborate features	0.79 ± 0.02
Logistic regression, SSVM refinement, elaborate features	0.68 ± 0.02
U-net	0.9

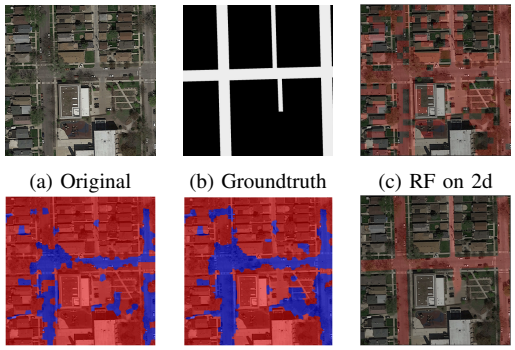
Table III: Concentrated results

it's groundtruth, and the predictions of all the different methods.

VII. Conclusions

To conclude, we can see that the road segmentation problem can be a challenging one. As with many cases nowadays, CNNs prove to be valuable tools to tackle this issue, since they learn useful features and

[3]Note that for the CNN case only one fold was run due to time contstraints. The reported F-score is on the validation set, and is calculated on a pixel level, contrary to the previous methods that are evaluated on a patch level, and on a superpixel level.



(a) Original (b) Groundtruth (c) RF on 2d

(d) RF on feature vectors of size 65 (e) Refined random forest (f) Unet CNN [Submission]

Figure 7: Example of predictions on unseen image (apologies for the different coloring).

provide meaningful representations of the underlying data structure.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[2] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 670–677, Sept 2009.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[4] Y. Lv, G. Wang, and X. Hu, "Machine Learning Based Road Detection from High Resolution Imagery," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 891–898, June 2016.

[5] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[6] https://en.wikipedia.org/wiki/Taxicab_geometry.

[7] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *Journal of Machine Learning Research*, vol. 6, no. Sep, pp. 1453–1484, 2005.

[8] A. C. Müller and S. Behnke, "Pystruct: learning structured prediction in python.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2055–2060, 2014.

[9] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *arXiv preprint arXiv:1505.04597*, 2015.

[10] F. Chollet, "Keras." https://github.com/fchollet/keras, 2015.