Project 2: Road Segmentation

Team Bazinga: Gael Moccand, Pascal Bienz
gael.moccand@gmail.com, pascal.bienz@gmail.com
Machine Learning Course, EPFL

***Abstract*—In this work we present different methods to segment roads on satellite images. We first show how we can augment an image dataset when the one at disposal is too small to properly train a machine learning algorithm. Then we quickly demonstrate what features can be exploited and how to handle them in order to make the best prediction with a linear logistic regression. Finally, we present a method based on a deep fully convolutional neural network architecture for semantic pixel-wise segmentation called SegNet.**

I. Introduction

The goal of this work is to segment roads on satellite images (Figure 2) by using machine learning techniques. In other words, we want to assign a label (road or background) to each pixel of the image. Before selecting the best algorithm, an effort is made on how to augment a small image data set and how to get the most relevant features out of it. Then we present 2 different class of algorithm. The first one is a linear logistic regression whereas the second one, called SegNet [1] uses a more complicated scheme based on a convolutional neural network (CNN).

A set of $N = 100$ training images of size 400×400 pixels is provided. The set contains different aerial pictures of different urban areas. Together with this traing set, the corresponding ground truth grayscale images (Figure **??** are also available. Note that the ground truth images need to be converted into label images. Concretely, each pixel y_i can only take one of the two possible values corresponding to the classes: road label $(y_i = 1)$ or background label $(y_i = 0)$. In order to binarize the ground truth images, a threshold of 25% is set. This means that every pixel with an intensity below than 75% of the maximum possible values is set to 0. With 8 bits images, the maximum value is 255 which sets the threshold to 191. This pixel threshold has a direct impact on the width of the road label in the computed label image.

The problem that arises with such a small training set (100 images only) is overfitting. Moreover in order to train any convolutional neural network properly it is necessary to augment the dataset. Analysing the training set, it is obvious that it contains mainly pictures with strictly vertical and horizontal roads. For that reason, creating new images by rotating the original ones allows to increase the size of the dataset and generates data which will be useful to better train the algorithm. Specifically, we rotate each image by angles going from 5 to 355 degrees every 5 degrees (i.e. 5, 10, 15,..., 355). That way we generate a set of images with roads in every directions. In summary, for each image of the original training set, 70 images are generated using

Figure 1. Example of satellite image.

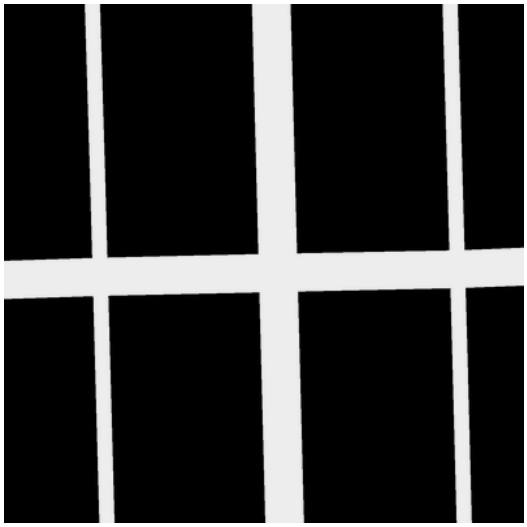

Figure 2. Ground truth of satellite image example.

the rotations, resulting of a new training dataset of 7100 images. This augmented training set is then suitable for training different CNN.

II. Methodology

In order to gain computational efficiency, square patches can be used instead of working with every pixels (see Figure 3). This make sense because a road is never composed by a single pixel but is rather made of blocks of pixels. The smaller the patch, the longer the simulations, but the finer the prediction. It is therefore important to find a trade-off. We've found that taking patches of size 8×8 gives decent result in a reasonable time. Within each patch, the mean and the variance in the 3 channels (RGB) are computed. On top of these 6 features, we add the computation of the histogram of oriented gradients (HOG) in 8 directions. The HOG is a descriptor used in many computer vision tasks for object detection purpose. It also consists of splitting the image in patches and gives their gradient orientation quantized by the angle and weighted by the magnitude. This makes a total of 14 features per patch. Since we have $50 \times 50 = 2500$ patches, it makes a total of 35000 features per image.
.

Figure 3. Example of a prediction using 16×16 patches. The predicted road are in red. Each red square correspond to a patch

The feature matrix is pretty sparse like shown on Figure 4. The histogram shows a large peak of zero followed by a decay. This decay-like shape suggested us to manipulate the features in order to get a distribution following a normal distribution. This can be obtained by taking the square root of the features and can be observed on Figure 5. .
.

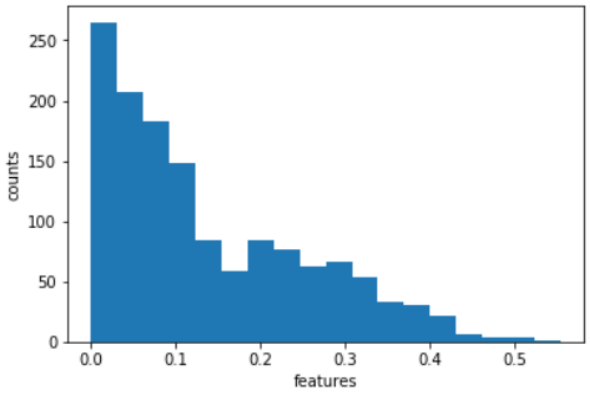

Figure 4. Histogram of the features computed on one of the satellite image

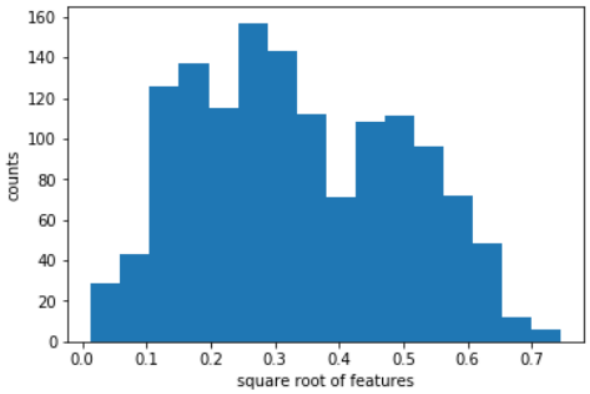

Figure 5. Histogram of the square root of the features computed on the same image

In order to apply a cross-validation, the training set is split into 2 parts. 80% is used for training (5680 images) and 20% (1420 images) for testing.

These features are fed to a simple linear logistic regression using scikit-learn. To compare the methods, we compute the percentage of accurate patch prediction using the testing test.

As a second step, we use the SegNet architecture which is a deep fully convolutional neural network. The SegNet architecture consists of a sequence of non-linear processing layers (encoders) and a corresponding set of decoders followed by a pixelwise classifier. Typically, each encoder consists of one or more convolutional layers with batch normalisation and a rectified linear unite (ReLU) non-linearity, followed by non-overlapping maxpooling and sub-sampling [2]. The Figure 6 shows the SegNet overall architecture.
Segmentation problems often use spatial softmax to try to classify each pixel. The Loss $\mathcal{L}$ used in SegNet is basically a Spatial Multinomial Cross-Entropy that runs on each pixel of your output tensor, comparing with your label image.
.

The SegNet implementation in tensorflow is taken from github reference code [3]. Two versions of SegNet are used

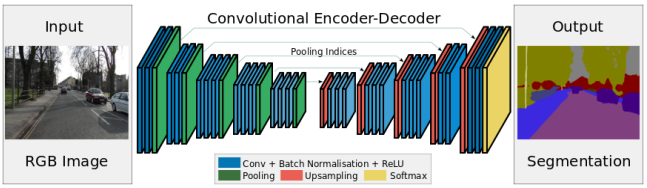

Figure 6. SegNet architecture [2].

with the reference code, "connected" and "gate connected". Both versions take the output of the previous convolver and the output convolver of the corresponding Encoder part as input for the Decoder.

Again, the model is trained with the augmented training set of $N = 7100$ images and the corresponding label images with a mini batch of 50 images. The initial learning rate is set to 0.001 with a decay every 10000 steps. Note also that the size of the image is reduced to 224×224 pixels in order to speed up the training of the neural network.

This time, the accuracy of the method is evaluated using the label of each pixel of the images of the testing set.

III. Results

The first method using the linear logistic regression has been dropped pretty fast because even with the features transformation (taking the square root), the prediction didn't exceed 0.59 on the testing set. Taking into account that prediction by flipping a coin would give 0.5, it is not a big achievement. This is probably due to the fact that the mean, variance and HOG are not sufficient to differentiate the roads from the rest of the objects.

Regarding SegNet, the results are way more encouraging. Indeed the prediction rate almost reaches 0.9, witch both SegNet connected (0.86) and connected-gate (0.87).

Method	Prediction rate
Patch + logistic	0.59
SegNet connected	0.86
SegNet connected gate	0.87

Table I
Summary of prediction rate of different methods.

IV. Discussion

The logistic regression yields pretty disappointing results. This can be explained by the reasons given above and by the fact that using patches has a main drawback. With this method we loose the continuity of the image and thus the information of the neighbor pixels at the boundaries of the patches. For instance, since a road is continuous, there is a bigger chance that a pixel is a road if its neighbors are roads than if they are part of the background. With more time, it would be interesting to keep the same prediction method but use much more features and possibly get information on the neighbor patches.

But finding the most relevant features that would lead to a more accurate with the same training algorithm is a tedious job. For this reason we decided to use a deep learning method instead.

In the case of SegNet, it actually overfits a bit. This can be observed on Figure 9 which shows that after 7 iterations, the spatial loss increases again

Figure 7. Complex example of satellite image

Figure 8. Prediction of the complex example using SegNet

V. Summary

In this work we have shown that one can have a fairly good prediction using SegNet.

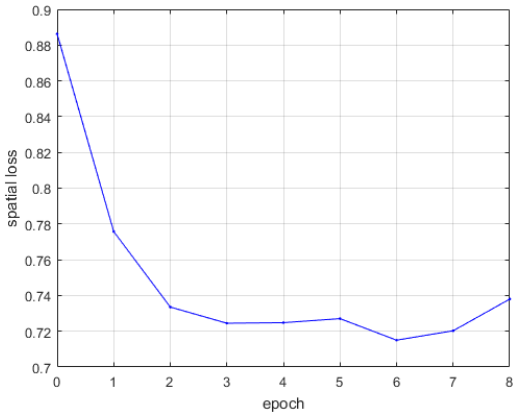

Figure 9. Spatial loss w.r.t the epoch number. It overfits after 6 epochs.

REFERENCES

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *CoRR*, vol. abs/1511.00561, 2015. [Online]. Available: http://arxiv.org/abs/1511.00561

[2] V. Badrinarayanan, A. Handa, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling," *ArXiv e-prints*, May 2015.

[3] L. Araujosantos, "Learn Segmentation," https://github.com/leonardoaraujosantos/LearnSegmentation, 2017.