Project-II by Group 28: B. pseudomallei

Juraj Korcek & *Christian Tresch* & *Pierre Colombo*

January 22, 2017

Abstract

In this report we summarize our findings for the Project-II done for Pattern Classification and Machine Learning course taking place during winter semester 2016 at EPFL. The aim of this project is to train a classifier to segment roads in satellite pictures. The data consists of two parts: a set of RGB images acquired from GoogleMaps and ground-truth images where each pixel is labelled as road or background. The amount of images available for training is 100. It is quite limited amount and therefore every image is split into several patches in order to increase the number of samples available for training. We have observed and analysed given dataset in order to get information about features that would help us classify the data the best. We have tried different preprocessing steps by applying various filters and transformations to generate additional samples and features. As it is current state-of-the-art for given problem, we have used a Convolutional Neural Network (CNN) to process those features.[1] Additionally, we have done crossvalidation to optimize number of epochs of the network and the patch size. In the end, we have applied postprocessing to denoise the output of the network. We get an accuracy of 87% at the end.

1 Introduction

In this paper we describe a thought process that lead to our solution. The data preprocessing phase has been one of the most important phase of our project. Our objective has been to apply various image processing filters and transformations to the original dataset in order to create new samples and features that would improve the performance of the classifier. We have chosen to use a Convolutional Neural Network (CNN) to tackle this segmentation problem. After the classification done by the CNN we have added a postprocessing step (using a filter again) leveraging the structure of the roads.

2 First Observations

By looking at the dataset provided we have noticed some geometric properties about the pictures and the roads:

- all the pictures have approximately the same features (luminosity, contrast, colors) and that is why we have chosen not to normalize the dataset by subtracting the mean and the variance
- the road texture and color are probably very important features for the classification
- the roads are composed of a connected pixels, i.e. there is no part of non-road in the road (useful for the postprocessing step)
- the edges are very important to determine the limits of the road

3 Choice of algorithm & Structure

We have decided to use a CNN because, currently, it is state-of-the-art for image segmentation problems such as this one.[1]. We have used TensorFlow library to implement the network. Our CNN has following structure: an input composed of 3 or 5 channels, two convolutional layers and a pooling layer with a soft-max loss as the activation function. The layout scheme is depicted by Figure 1.

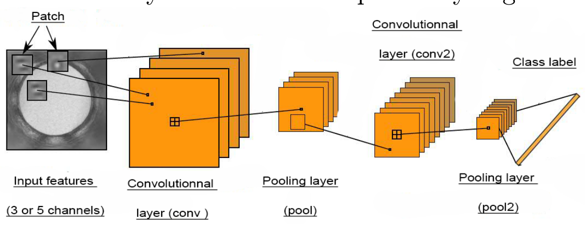
Patch
Convolutionnal layer (conv2)
Class label
Input features (3 or 5 channels)
Convolutionnal layer (conv)
Pooling layer (pool)
Pooling layer (pool2)

Fig. 1: *Network layout*

4 Data preprocessing

Our preprocessing consists of 2 parts: the first one is an augmentation of the dataset by applying various transformations to the original images and the second one is creation of new features by using different image processing filters .

4.1 Data augmentation

In this part we explain the techniques used to increase the size of the dataset.

4.1.1 Rotation

In order to obtain more training samples we have decided to rotate every image 3 times (by 90, 180 and 270 degrees). We can get 300 more images from the 100 original ones this way and thus have 400 images to train on altogether. We have done the same for all the corresponding groundtruth images.

4.1.2 Flipping

After rotation, we have flipped every rotated image horizontally. This way we have created another 400 training images. Afterwards, we have flipped every rotated image vertically in order to get additional 400 training images. Again, we have done the same for all the corresponding groundtruth images.

4.2 Features engineering

The CNN chooses itself what features are useful. That is why we have tried to add several features and let the CNN choose the ones it prefers. In the following subsections we discuss all the features that we have experimented with.

4.2.1 RGB picture

The RGB picture carries a lot of information about the the texture and the color of the road. This corresponds to the original image. An example of one such image can be seen in Fig. 2.

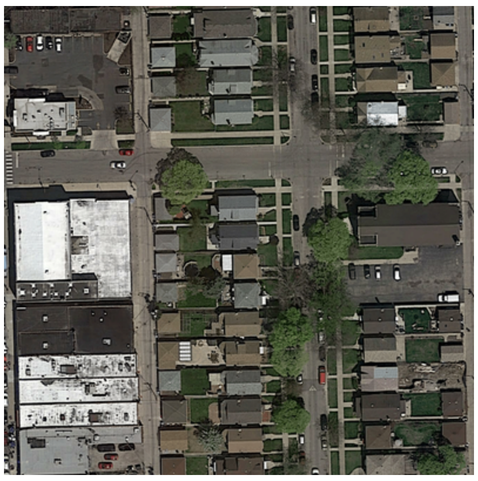

Fig. 2: *Original image*

4.2.2 Grayscale Picture

The grayscale picture is obtained from a linear transform of the RGB picture. This picture carries some information, but we assume that the CNN can do the transform itself if necessary and thus we have finally decided not to use this feature.

Fig. 3: *Grayscale image*

4.2.3 Edge detection

When a human tries to classify pixels into road and background classes the road's edges are very useful.

- **Canny edge detection**: The Canny filter is widely used for edge detection. The canny edge detector first smooths the image to eliminate a noise. Then, it finds image gradient to highlight regions with high spatial derivatives. Another big advantage is that we do not need to set the parameters very precisely. The output of the filter is a binary picture. In Fig. 4 the effect of application of the Canny filter on grayscale image is shown.
- **Sobel edge detection**: The Sobel operator performs a 2D spatial gradient measurement on an image and thus it emphasizes regions of high spatial frequency that correspond to edges. The result of application of this filter to grayscale image can be seen in Fig. 5.

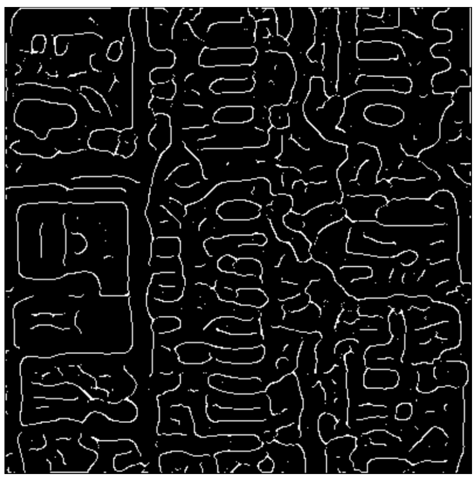

Fig. 4: *Canny filter on grayscale image*

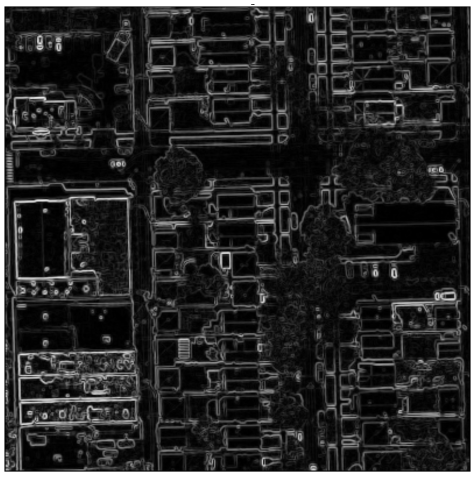

Fig. 5: *Sobel filter on grayscale image*

4.3 Summary

By doing data augmentation we have increased the size of the training dataset from 100 to 1200 images. When it comes to feature engineering, after evaluating performance of each filter we have decided to keep only Edge Detection with Canny and Sobel because they are the only two that bring tangible prediction accuracy boost.

5 Improvement of the CNN & Parameter Optimisation

In this part we describe the improvements we have done to the CNN. We have chosen not to change the structure because we have not found general ideas ruling the behavior of a CNN given its structure.

5.1 Sliding window approach

5.1.1 Creating the patches

In order to attain a better resolution for the patch predictions we have decided to use a sliding window approach instead of the original patch-tiling implemented in the sample script. Here we are able to define $CROP\_STEP$ variable that defines the step-size of the window sliding over the image generating many more patches than with the initial solution.

We have adapted the given code for CNN such that the patches generated from the image are not mutually exclusive anymore. Instead, we generate a patch starting at every n-th (n = $CROP\_STEP$) pixel (except for the cases where the patch would reach outside of the original image). In other words, two consecutive patches (in horizontal sense) overlap everywhere except for n (n = $CROP\_STEP$) columns of pixels. This way we get many more training samples. For example, given an input image of size 400 x 400px and default patch size of 16 x 16px with original code we would have obtained (400 / 16) * (400 / 16) = 625 patches per image With our approach and a step size of 8 we get (400 / 8 - 1) * (400 / 8 - 1) = 2401 patches per image. This allows for a much more finegrained but also results in overlapping prediction of the patches, which needed according post-processing discussed in the follwing section.

5.1.2 Deconvolution of patches

Due to this overlaps we additionally implemented a 'Deconvolution' function that reconstructed the fraction of label predictions for the final output image. This was achieved by summing the label values for each patch on top of each corresponding image section and then subsequently averaging the value by the amount of label values received for that pixel. Consequently, we obtained a fractional value for the label prediction. This was not a problem but rather an advantage since it allowed the subsequent *mask_to_submission* algorithm to have a more distinct perception of the actual value in an area when it finally applied the labels for the individual 16x16 patches which were ultimately binary labels.

5.2 Parameter optimization

We have decided to do crossvalidation in order to set two hyperparameters - the number of epochs of the neural network and the optimal size of patch. We have experimented with number of epochs ranging from 4 to 8. When it comes to the patch size, we have tried sizes between 7 x 7px and 40 x 40px. We have obtained learning curves shown in Fig. 8 and Fig. 7 respectively. Using the results we have determined that the optimal number of epochs for our network is 7 and the optimal patch size is 16 x 16px.

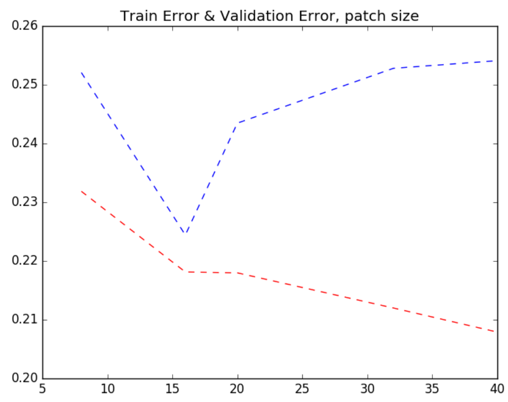

Fig. 6: Learning curve for patch sizes from 7x7px up to 40x40px

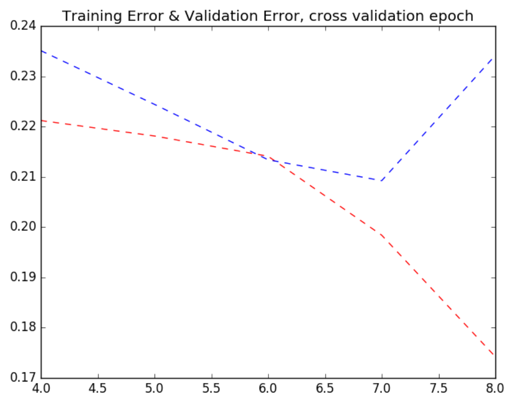

Fig. 7: Learning curve for epochs ranging from 4 to 8

5.3 Avoiding overfitting: Dropout layer

In order to avoid overfitting we have implemented a single dropout layer. We have set a dropout of 0.5 % during training only. Indeed dropout prevents units from co-adapting too much by randomly dropping units.

5.4 Threshold Optimisation

The threshold is the parameter that determines whether the patch is considered to be a road or not. For instance, by setting it to 0.5, if more than half of the patch is road we consider the patch to be a road; otherwise it is background. We have ajusted this parameter manually. We got the best results by setting the threshold to 0.60.

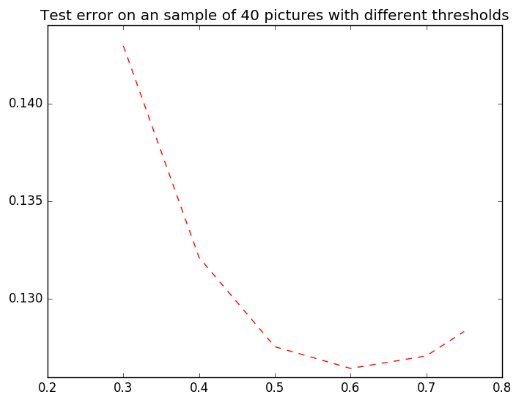

Fig. 8: Threshold optimisation

6 Postprocessing

At the end, in order to slightly improve the classification results we have implemented postprocessing.

Our postprocessing consists of running opening filter (erosion & dilation) on the output of the neural network. This effectively removes the noise from the roads. This is shown in Figure 9.

For this filter we have to set a parameter which is the POSTPROCESSING_DISKSIZE. It is linked to the biggest region (black or white region we can be absorb by applying the postprocessing part. After several trials we have set POSTPROCESSING_DISKSIZE to 6.

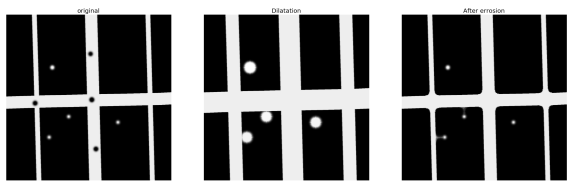

Fig. 9: *Post Processing (form left to right Input picture, dilatation, erosion*

7 Results

In this part we explain and compare the different neural networks that we have trained and tested using the different features that we have prepared.

7.0.1 Baseline Model

The first neural network we have deployed was the simplest one. We have taken the 100 RGB pictures to train the CNN, no parameters optimisation. In the following table we show the different parameters of our pipeline.

Parameter	Value
Epochs	5
Batch Size	16
Patch size	16
Threshold	0,25
Number of Channels	3
Input	100 RGB images

With this basic implementation we get a score of 81% on the training set and, according to Kaggle, 77% for the test set.

7.0.2 Second Model

In the next step we have plugged in the edge detection filters outuput into input of the CNN. We have kept the same structure and the same parameters as before.

Parameter	Value
Epochs	5
Batch Size	16
Patch size	16
Threshold	0,25
Number of Channels	5
Input 1,2,3	100 RGB images
Input 4	Canny edge detection
Input 5	Sobel edge detection

With this basic implementation we get a score of 83% on the training set and, according Kaggle, 80% for the test set.

At this point we have noticed that the edge detection brings a small amount of information

We have decided to change the threshold in the decision part. Indeed, by changing the threshold from 0.25 to 0.5 we went from 80% to 84%.

7.0.3 Final Model

In the final model we have tried to bring all our results together (hyper parameters optimisation, features engineering & prepossessing, threshold optimisation, sliding windows, postprocessing).

However, we have encountered computational issues. With computers and even with servers we were not able to train a CNN with 5 channels (without the grayscaled picture) and 1200 input samples because of the memory it requires.

The best results we have obtained are for the pipeline with the following parameters:

Parameter	Value
Epochs	7
Batch Size	16
Patch size	16
Threshold	0,5
Number of Channels	3
Input 1,2,3	800 RGB images
Windows	Sliding windows
DISKSIZE	6

The result we have obtained is 87.055% in the test set.

8 Summary

In this paper we have provided a quite efficient way to deal with satellite images and road segmentation. Our algorithm using a TensorFlow CNN has reached a score of 87%, this is an more than an acceptable accuracy.

9 Discussion

During all our research we have faced a big problem: we have lacked the computational power (memory in particular). We have not been able to put all our results together. We think that with more computational power and by training the CNN with 1200 pictures and 5 channels we could have increased our score conisderably. These issues are the reason why we have not been able to use a big part of our feature engineering step (Sobel & Canny filters).

In addition we list some ideas we di not have enough time to implement:

- Increased PATCH_SIZE in combination with decreased CROP_STEP: An increase in the patch size might have allowed for more information to be processed within the vicinity of a single road/no-road label patch. Since both of these actions, decrease of crop step and increase of patch size, had significantly increased the computational power needed and exceeding the hardware and time limits available we refrained from those attempts.
- Improving the post-processing step: training a new neural network for the post processing part, taking as input the predictions and the groundtruths during the training part, this probably would be a more efficient post-processing algorithm.

10 Acknowledgements

We are thankful to Aurelien Lucchi from ETH Zürich who has provided us with an example of implementation of a CNN using TensorFlow library that has helped us considerably throughout the project.

References

[1] CNN-AWARE BINARY MAP FOR GENERAL SEMANTIC SEGMENTATION from Mahdyar Ravanbakhsh & Hossein Mousavi & Moin Nabi & Mohammad Rastegari & Carlo Regazzoni at https://arxiv.org/pdf/1609.09220.pdf