Project 2 : Road Segmentation

Team : *All Roads Lead 2 Segmentation*
Clément Lefebvre, Vincent Pollet, Aymeric Galan
Computational Sciences & Engineering, EPFL, Switzerland

***Abstract*—In this report we explore the problem of road segmentation using Deep Learning techniques. This is a recurrent problem [1], [2] in the field of Deep Learning because the results can be generalized for other segmentations problem using more classes. Our solution was based on a Convolutional Neural Networks (CNN) architecture where the data given to our model was each image divided in patches. More particularly, we show the excellent generalization properties of CNN by using transfer learning to obtain our best result of 94.518% F1-score.**

I. Introduction

The challenge that we tackled is to classify 16×16 patches of an aerial image between "road" or "background". The provided dataset contains 100 images of size 400×400 (RGB) and corresponding groundtruth masks. The test set is composed of 50 images of size 608×608. We first present the techniques used to enrich the dataset (image augmentation), then a deep learning model using a convolutional neural network trained on the provided dataset using the Keras library [3] with TensorFlow [4] back-end. We then use the well known VGG16 [5] architecture, training it from scratch on our dataset. Finally, our best result was obtained by using a VGG16 model pre-trained on the `ImageNet` dataset, on top of which was plugged a classifier.

II. Data augmentation

As a preliminary study, we propose in this section to get a sense of the influence of three factors : the training size, the data augmentation, and the number of epochs. The second one is closely related to the dataset size, since it is a way of increasing the dataset size.

We increase the dataset size by applying the following transformations to original training images : rotation, zoom, and horizontal and vertical mirror flips.

The dataset initially contains 100 images. The transformations are applied randomly to generate 1 500 extra images from the initial set. In the following results of this section, when the training size is 100 or below, images from the original non-augmented dataset are used.

In order to provide a general comparison, which does not depend on the peculiar models we built for this project, we choose to train the basic CNN model provided in the machine learning course as an example. It is implemented using the TensorFlow framework.

In order to quantify the influence of training epochs, we compute the mean F1 score for training and validation datasets, based on a 10-fold cross-validation. Results obtained with 1, 5 and 10 training epochs are shown on Fig. 1. A peak can be seen for a size of 100 images which corresponds to the 100 original images with no data augmentation. This way, the contrast between with the slightly augmented dataset, besides the size of the training dataset itself, is emphasized. Indeed the F1 score drops dramatically when the dataset size is doubled from original images and augmented images. As the number of augmented images further increases, the score increases again to finally reach a F1 score better than the one obtained with the whole original dataset. Hence it suggests that image augmentation really improves the robustness of the neural network predictions. Going from 1 to 5 epochs appears to improve a lot training and validation scores, but increasing further the number of epochs causes only a small change in final result. These results suggest the existence of a value from which the performance does not increase by much with further training. This value could vary much from a model to another though.

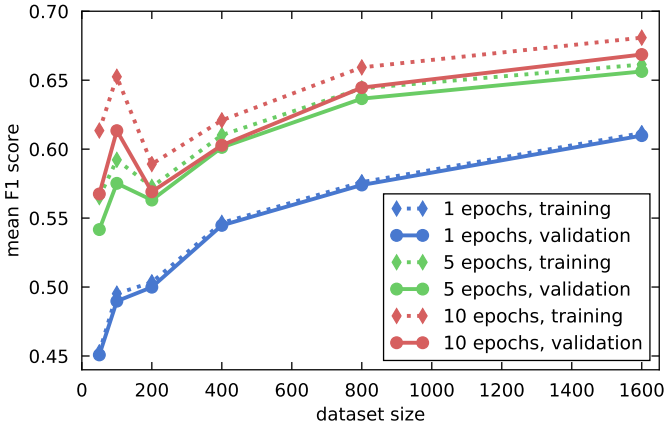

Fig. 1. Micro F1 score averaged over a 10-fold cross-validation obtained with the CNN provided with the project. It is plotted as a function of the training dataset size, for various numbers of training epochs. The original size is 100 images, and has been augmented using rotation, zoom, and flip operations to reach a size of 1600 images.

Results of the same kind were obtained by varying this time the type of image augmentation. Three datasets are constructed from the base one, using the following geometrical transformations : only rotations, zoom operations added to rotations, and shear added to both previous operations. Results are displayed on Fig. 2. We observe that the type of image augmentation plays a role in the cross-validation performance. The addition of zoom operations do not improve significantly resulting F1 scores. On the contrary, adding shear operations appears to bring useful informations to train the model. The performance

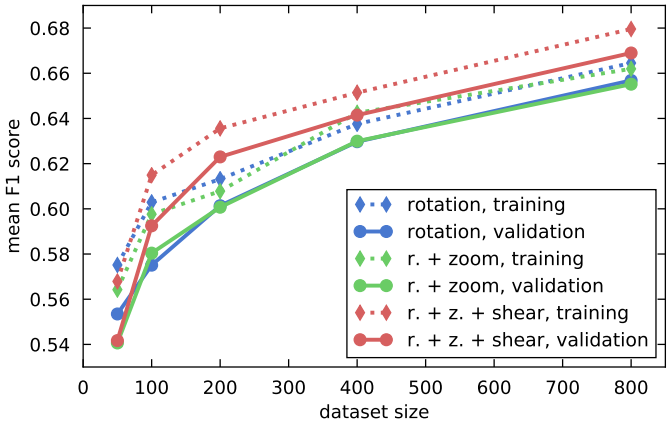

Fig. 2. F1 score averaged over a 10-fold cross-validation obtained with the CNN provided with the project. It is plotted as a function of the training dataset size, for various types of images augmentation. The original size is 100 images, and has been augmented using rotation, rotation+zoom, and rotation+zoom+shear operations to reach a size of 800 images.

of this basic CNN model is quite low. For this reason, in the remaining part of this report, we concentrate on different CNN models implemented to improve a lot predictions, using this knowledge gained from these results to augment the data.

III. Input data

After the data augmentation, we had to decide how our CNN would be fed with the data. The size of the images didn't allow us to feed them directly together with the groundtruths.

A. Patch division

We opted to divide the images in patches of size 16×16 and feed the patches to our CNN. With 100 training images of size 400×400, we get 14 745 600 different patches possible.

B. Groundtruth label

To each patch we assign a label by taking the mean of the groundtruth patch and applying a threshold function (see (1)).

$$f(n) = \begin{cases} 1, & \text{if } n \geq 0.25 \\ 0, & \text{if } n \leq 0.25 \end{cases} \quad (1)$$

where n is the mean value of the groundtruth patch. We chose to use a threshold of 0.25 to give more weight to the road as they are less present in the dataset.

C. Window contextualization

A good strategy is to use *window contextualization* [6] on the image, that is a part of the image centered on the patch and including its surroundings, as shown on Fig. 3. This technique allows us to give more input features to the CNN to make the prediction, which leads to better results than feeding the patch only. However, it is computationally more challenging since a larger input image implies more computation for a given CNN.

Besides, this technique requires to pad the image, otherwise a window centered on a patch at the edge of an image would not be defined. A "reflect" (mirror) padding was used for the results presented in this report to expand the image in the most realistic way. Another solution would have been to constrain the window to be entirely contained in the image, however this would have considerably reduced the number of possible positions for the patch, hence reducing the already small dataset. Fig. 3 shows resulting mirror boundary conditions as well as a patch and its context window.

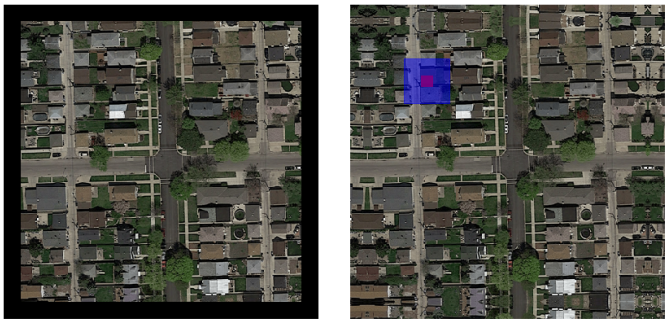

Fig. 3. Left : original training image (black borders were added for the comparison). Right : mirror boundary conditions applied to the left image. The 16×16 patch (in red) is displayed within its 64×64 context window.

D. Image augmentation

The initial training dataset contains 100 images, which is a quite small number. In order to increase performance of the models developed for this work (II), we augment the training dataset by applying the following transformations :

- rotation
- shear
- horizontal and vertical flip

As seen before, in order to avoid unwanted zoom operations after applying operations we perform a mirror padding. An example of a training image / groundtruth before and after a rotation with mirror boundary conditions is shown on Fig. 4.

IV. Models

We tried three different approach to solve the problem of road segmentation. First, we tried a model with 4 convolution blocks each containing 2 convolutions with kernel size 5×5 and convolution depths : 64, 64 ; 128, 128 ; 256, 256 ; 512, 512. The second model is based on the VGG (Visual Geometry Group) model [5] with 13 convolutional layers, we first tried to train this model from scratch after adding our own classifier at the output of the model (see table I) and then we tried a second training by loading the pre-trained VGG16 model using the weights from the `ImageNet` competition and tuning our classifier on top and then fine-tuning the last block of the VGG architecture.

V. Training

Here we present the different techniques used to train our models :

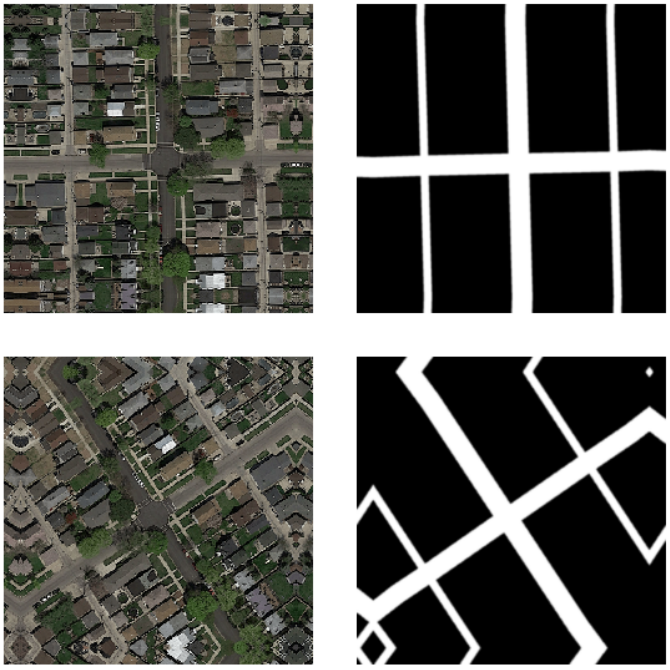

Fig. 4. Example training image and its groundtruth counterpart before and after a rotation operation applied for data augmentation.

Layers	Details	frozen weights model **I**	frozen weights model **V**
Input (patch)	$64 \times 64 \times 3$		
Convolution 1 + ReLU	64 3×3 filters		
Convolution 2 + ReLU	64 3×3 filters	yes	no
Max Pooling	2×2		
Convolution 3 + ReLU	128 3×3 filters		
Convolution 4 + ReLU	128 3×3 filters	yes	no
Max Pooling	2×2		
Convolution 5 + ReLU	256 3×3 filters		
Convolution 6 + ReLU	256 3×3 filters	yes	no
Convolution 7 + ReLU	256 3×3 filters		
Max Pooling	2×2		
Convolution 8 + ReLU	512 3×3 filters		
Convolution 9 + ReLU	512 3×3 filters	yes	no
Convolution 10 + ReLU	512 3×3 filters		
Max Pooling	2×2		
Convolution 11 + ReLU	512 3×3 filters		
Convolution 12 + ReLU	512 3×3 filters	no	no
Convolution 13 + ReLU	512 3×3 filters		
Max Pooling	2×2		
Flatten	2048		
Dense 1 + ReLU	256 + L2-reg. ($\lambda = 10^{-4}$)	no	no
Dropout	$p = 0.5$		
Dense 2 + SoftMax	2		

TABLE I
Layers configuration of our models **I** and **V** developed for this project (based on ConvNet D from Table 1 of [5]). A block is mentioned as "frozen" when its parameters are non-trainable (fixed values).

1) Optimizer : We used the Adam optimizer [7] which is computationally efficient and has little memory requirements. The starting learning rate is 10^{-4} and is divided by 5 when the validation F1-score stagnates during 2 epochs.
2) Activation functions : we used rectified linear unit (ReLU) for the convolutional layers and a SoftMax for the classifier.
3) Weight initialization : a Glorot uniform distribution [8] is used to randomly initialize the weights of our kernels.
4) Regularization and Dropout : to avoid overfitting to our training dataset, we used a L2 regularization on the classifier and added Dropout layers [9] to avoid one unit relying on other units.

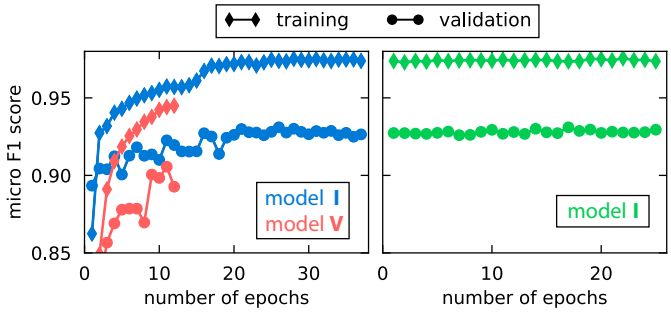

Fig. 5. Micro F1 score obtained for the models **I** and **V**. The left plot shows the training of the model **V** and the training of the classifier of the model **I** where the 5^{th} convolutional block is frozen. The right plot shows the second part of the training with the last 3 layers of the VGG architecture unfrozen.

VI. Results

The CNN **S** model has been used as a first model to assess the quality of the augmented dataset and the effectiveness of the window contextualization technique. It reached a maximal score of 90.072% on the `Kaggle` competition (cf table II). A visualization of the filters and their activation on a typical image window are displayed on figure 6. An important number of filters are inactive, which is limiting the performances of the model.

The CNN **V** model performed better reaching 92.210% of F1 score ; the learning curves are displayed on Fig. 5. The model overfits the training dataset after a few epochs, which is difficult to prevent due to the model important complexity and the small quantity of training data (even with image augmentation).

To prevent the overfitting when using the CNN **V** model, we used the CNN **I**, which has the same architecture but the weights of the first layers are frozen (untrainable). This model has been trained on a much bigger dataset, which results in very smooth filters as can be seen in the visualization of the filters of the first layer (Fig. 6). The first layers of a CNN encodes general features like color, or edges. The combination of these quality filters with the fine tuning of the relatively few weights of the last layers of the network allowed us to reach a score of 94.518%.

Scores obtained with our different models are shown in table II. A prediction of roads by the model **I** on one of the test images is shown on Fig. 7.

model	CNN **I**	CNN **V**	CNN **S**
F1-Score	94.518%	92.210%	90.072%

TABLE II
Scores of our different models in the `Kaggle` competition.

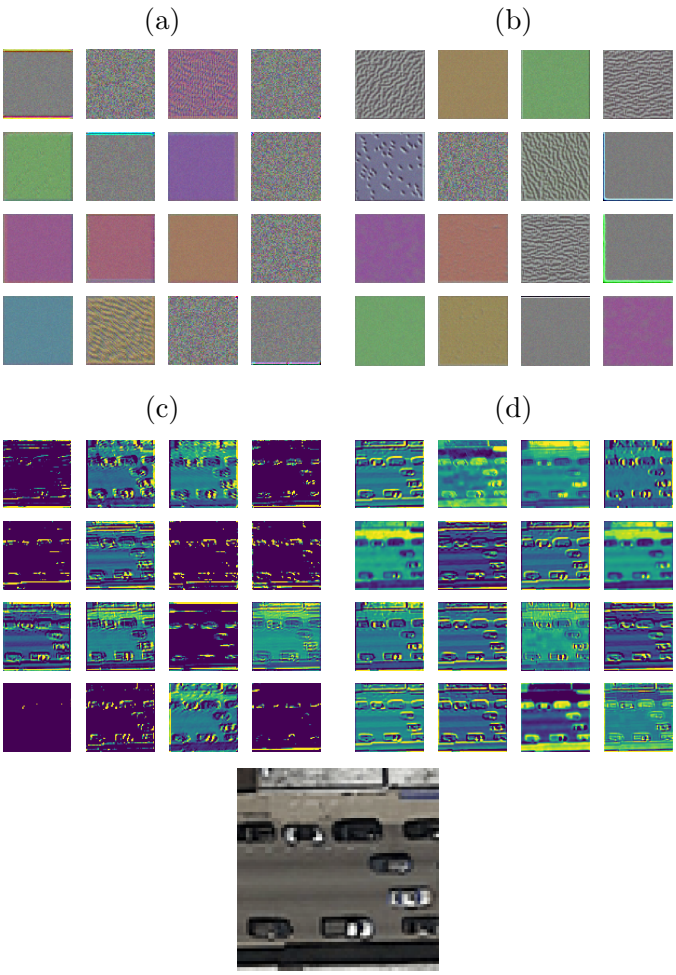

Fig. 6. On the first row: visualization of the weights of 16 kernels of the first convolutional layer of the CNN **S** (a) and CNN **I** (b). An important number of the filters of the CNN **S** model show small noisy outputs: these are dead filters. Comparatively, the filters of the model CNN **I** are all active. On the second row: (c), respectively (d), shows the activation maps of the filters in (a), respectively (b). The input of the filters is the image window at the bottom. The filters of the CNN **I** model detect much more sharp feature in the image.

VII. Summary

With this study, we showed the efficiency of Convolutional Neural Networks for pixel classification. We obtained a F1-score of 94.518% using transfer learning with the VGG16 architecture pre-trained on the `ImageNet` dataset. This shows that the learned features of a CNN are transferable to other image classification tasks. There are multiple ways to improve our model : a more comprehensive dataset could lead to significant improvements of the results as we showed in II. Moreover, a more thorough optimization of the hyper parameters can also lead to better results.

Fig. 7. Prediction example obtained with model **I**.

References

[1] P. Kaiser, J. D. Wegner, A. Lucchi, M. Jaggi, T. Hofmann, and K. Schindler, "Learning aerial image segmentation from online maps," 2017.

[2] M. Schwab, M. Karrenbach, and J. Claerbout, "Making scientific computations reproducible." [Online]. Available: https://www.ethz.ch/content/dam/ethz/special-interest/baug/igp/photogrammetry-remote-sensing-dam/documents/pdf/marmanis-isprs16.pdf

[3] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[4] M. Abadi, A. Agarwal, P. Barham, and e. a. Eugene Brevdo, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[6] F. Tschopp, "Efficient convolutional neural networks for pixelwise classification on heterogeneous hardware systems," 2015.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[8] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," 2010.

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, 2014.