Report machine learning project 2

Victor Le, João Silveira, Santiago Garcia Serrano
Department of Computer Science, EPFL Lausanne, Switzerland

***Abstract*—In this paper we explain the methodology used to solve the proposed road segmentation challenge. We discuss the different techniques used to pre-process the images to obtain better prediction results using convolutional neural networks (CNN) to find the road pixels in images.**

I. Introduction

The aim of this project is to develop a machine learning model to try to generate predictions of 16x16 pixel patches of images which contain road. The segmentation will be done on several RGB satellite images, and the goal that is tried to be accomplished is shown on Figure 1.

To achieve this objective, due to the low amount of data, data augmentation and image pre-processing techniques will be used to try to obtain meaningful and good quality information about the images. To obtain the predictions, convolutional neural networks will be used to categorize each patch as road or not road.

The rest of this report is organized as follows. We introduce dataset in Section II. The techniques used to augment data are presented in Section III. The transformations made to the images to improve results are shown in Section IV. The different models tried out are explained in detail in Section V. Road segmentation F1-scores results are given in Section VI. Finally, the summary and conclusions are presented in Section VII.

II. The dataset

We received two sets of data from the competition: the training set, which contains 100 color images of 400x400x3 with their respective groundtruth images, and the test set, 50 color images of 608x608x3. Since the patch size is 16x16, we get in total 62500 patches for the training set and 72200 for the test. After some exploratory analysis, it was found that the non-road patch are more frequent than the road one (46309 vs 16191), meaning that the dataset is unbalanced. To counter the unbalance, in the model that involved a CNN, we applied different weight for each class of road or non-road.

III. Data augmentation

The low amount of data given, one hundred examples, is not enough training data to achieve good results and, in addition, there exists a high risk to end up with overfitting. Fortunately, there is a way to generate more examples from the initial dataset. There are several transformations that can be used for images. However, the only valid transformations that produce similar images are mirroring and rotation. With 4 rotations per image and one mirror per rotation, we can obtain 8 new images from each original one, resulting in a total of 800 training images which translates to 5 million patches. See figure 2 as an example of the rotations and mirroring applied.

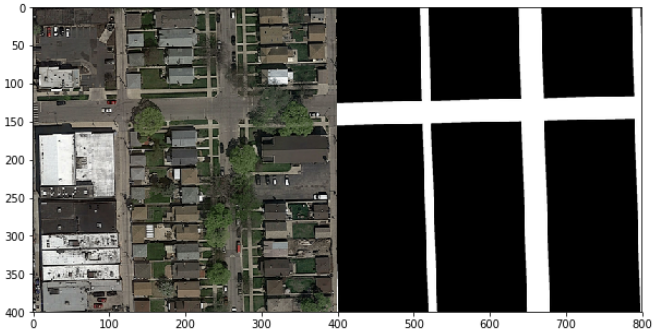

Fig. 1. A aerial image and its ground truth

Fig. 2. The result of the augmentation for one padded patch using rotation and flips

IV. Data transformations

As mentioned before, the patches needed to predict are of size 16x16 pixels. When only using such small patches, it is difficult to obtain any meaningful information to be able to identify correctly if each patch is road or not. To avoid the loss of information, a padding around the objective patch was added. The padding has size 24 pixels on all sides of the patch, increasing its size up to 64x64 pixels. With this larger patch, more information about the objective 16x16 patch can be inferred thanks to having more information available around the objective patch. An example of the transformation in figure 3 is shown. It shows a 16x16 patch and its 64x64 padded version. On the latter there is a red square which indicates the 16x16 patch we want to predict.

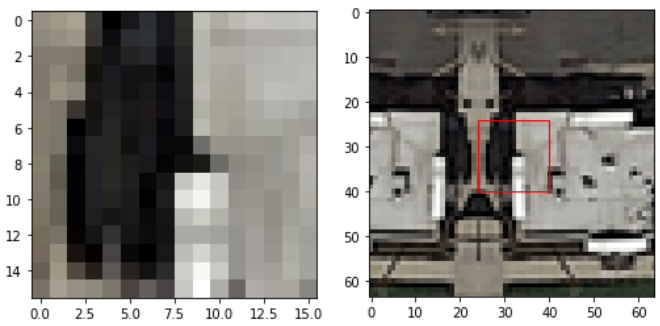

Fig. 3. A 16x16 patch and its 64x64 padded version

Fig. 4. The external mirrored padding of figure 1

The problem with this approach is how to add the external padding to the patches that belong to the edges of the original images. To be able to obtain it, it was first needed to also add a frame of 24 pixels to the original image, which was obtained by mirroring the 24 pixels around the edge, as shown on figure 4.

V. Models

In this section, the different models used are explained with greater detail. The pre-processing techniques for the different models are shown in the Jupyter notebook attached *Preprocessing.ipynb*. Moreover, there is a standalone notebook for each model with detailed steps.

A. Logistic regression

The first solution comes from the provided example code. It uses Logistic Regression for classification with the *liblinear* solver. The features extracted from each image are the average gray color and variance. The F1-score was the lowest since this method using these features is not optimized for image classification. A problem related to this approach is the reliability of manually extracting and selecting the features. It would be too time consuming and demands a deep understanding of the field to be able to select the right features. Even then, we could still be overlooking some of the features that adapt best for our specific classification. Thus we decided to drop this approach and proceed to using neural networks.

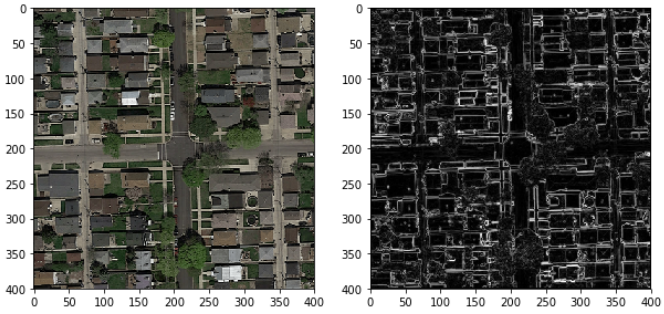

Fig. 5. Example of Scharr filter applied

B. Small patches + CNN

Our second solution does not classify the padded patches, but instead the raw 16x16 patches. We used the tensorflow code from the provided file *tf_aerial_images.py*. We trained the simple convolutional neural networks on the train set. The obtained accuracy was not very good. We realized that the 16x16 patch size is too small to get it classified. We then started by adding some padding like described in IV.

C. Segnet

The state of the art for image segmentation is with the SegNet [1] model. It uses one autoencoder to do the task. The model receives the aerial images and then outputs directly the road/non-road segmentation. We trained the SegNet-Basic for 300 epochs, and managed to get 0.92 accuracy on the train set. However, it was overfitting our training set, the segmentation for the test set and the validation set was completely false. Due to the high complications of optimizing a large scale net, the decision was to drop this model and continue with simpler ones.

D. Padded patches + CNN

Our fourth model is a standard CNN which receives as input the padded size 64x64 patch (III) and outputs the label road or non-road. The model is built using 3 convolutional max pooling and a final fully connected. We used 3x3 filters and 2x2 pooling size. The activation we choose is the Exponential Linear Units (ELU) which gives us the highest accuracy score [2]. We added after each pooling layer a dropout of 0.20 which helps to not overfit the data. The network was trained using the augmented training set and converged to a 0.92 f1-score after 150 epochs.

E. Scharr filter processing + CNN

An idea that came to mind to lighten the work of the neural networks was to pre-process the images to lower dimensions instead of giving the full RGB image. The first step was reading the image in black and white instead of color, reducing already the size by 3. This minimises the amount of features that every convolution has to cover. To try to make it even simpler, we also applied filtering to the input images. The scharr filter [3] is an optimization of the Sobel operator [4], that tries to find the gradients of the images, giving as output the edges of the shapes in the photo. An example of this filtering technique is shown on Figure 5 This reduces a lot the information the neural net has to process, increasing the training speed, and the result obtained from it ended up with an F1-score of 0.85698, not managing to improve over the previous result. This could be due to the Scharr filter being a 3x3 filter similar to the ones that the CNN already implements. If this filtering was useful for this segmentation task, one of the layers of the net would most likely apply it on its own, therefore not making it as useful as other methods to pre-process.

F. Padded patches + CNN + Denoiser

We realized that the final segmentation produced by re-assembling the corresponding patches was very noisy. So we used an autoencoder to denoise and smooth it. The denoiser receives the predicted segmentation image and outputs a denoised version. See figure 6. The autoencoder used has 2 convolutional max pooling layers for the encoding part and 2 upsampling layers for the decoding part. It was trained it directly using the output images generated by the CNN. The model was fast to train, it converged after a few epochs. The results improve from the model without the denoiser, as it manages to smooth the images, removing the isolated pixels and reconstructing broken road due to missing pixels. We get, at the end, a F1-score improvement of 0.00047. You can take a look to our notebook *CNN and Denoiser.ipynb*, it describes all the process.

G. ZCA preprocessing + CNN

In addition of the models presented above, a new idea for pre-processing the images was tried, but due to the long times taken for training the models could not be properly optimized, and it overfitted giving an F1-score 0.77864 from an accuracy on the training phase of over 0.94. The method applied is ZCA whitening[5]. The idea of this method is that the raw image data can be redundant since there exists a high correlation between adjacent pixels. The goal of this pre-processing strategy is to reduce the redundancy of the input images, obtaining as a result less correlated data, and similar variance features.

The algorithm implemented is like PCA, where only the most important features of the image are kept. It chooses a dimensionality reduction matrix where $R^tR = RR^t = I$. In this case, the matrix R chosen is U, which contains the eigenvectors of the covariance matrix of the images. The

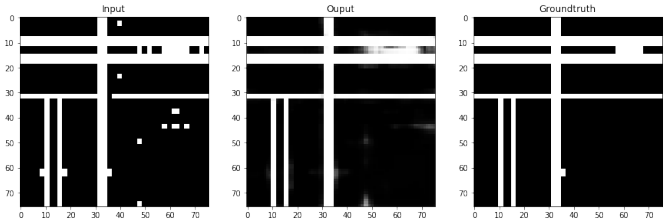

Fig. 6. The result of the denoiser. From the left to right : Input, Output and Ground truth

choice of this reduction matrix allows the whitened data tho be as close as possible to the original images. The advantage of ZCA against PCA is that in the end you keep all the original dimensions of the data.

For us to apply it, due to processing limitations, a whitened matrix was found for each batch of 3125 patches of the images, and then the average between the ZCA matrices was calculated and applied to all the patches. This was done so that all the training data and testing data would go through exactly the same pre-processing ZCA matrix.

Before finding the ZCA matrix, it is necessary that the data is zero centered, which is obtained by flattening the images and substracting the mean image by image. To finish, apply the L2-norm to the data to get image from the vectors of magnitude 1.

H. Fine tuning

Our last choice of model was to try to fine tune a pretrained model for image classification. We choose to use a DenseNet-121 which, although it is not the best model, it achieves a decent accuracy, having the advantage that it is not a large network, making it faster to train. It is trained on a large data set : ImageNet [6] (1.2M labeled images). So, the model has already learned lots of features regarding images, and captured universal features like curves and edges in its early layers. These are relevant and useful to most of the classification problems. However, back to our classification problem for road and non-road, the model took a lot of time to train, and has never given good result, predicting always non-road. Even being a small model compared to other state of the art models, it is too big for our available resources, we do not have enough computational power. Moreover we suspect that the aerial image is too different from the pre-trained dataset, and therefore the predictions were not as expected.

VI. Results

The results of each models are presented on the following table:

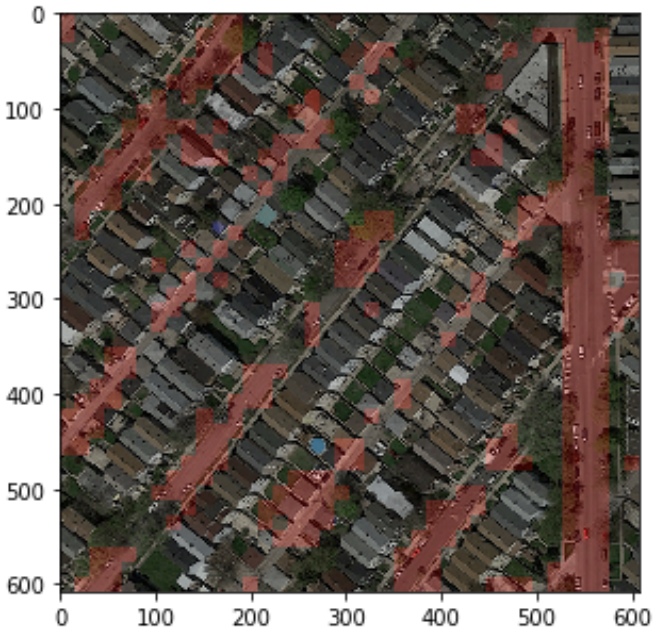

Fig. 7. A road segmentation from the best model : Padded patches + CNN + denoiser. The region with a red overlay means that the classifier predicted a 'road'.

Model	F1-score
Logistic regression (V-A)	0.71142
Small patches + CNN (V-B)	0.78124
Sharr filter + CNN (V-E)	0.85698
Segnet autoencoder (V-C)	0.81142
Padded patches + CNN (V-D)	0.92135
Padded patches + CNN + denoiser (V-F)	0.92182
ZCA preprocessing + CNN (V-G)	0.77864

As it can be observed on the table, the best results were obtained by using the padded patches from the images in RGB as inputs to the CNN, and denoising the output images by applying an autoencoder. The autoencoder helps to improve only sligthly, but this model, even without it, is the best option by a 0.06 in the F1-score. This is because it receives as inputs the patches with all the information from the different color layers, compared with other models used. You can see on figure 7 an example of a segmentation made in a image from the test set. The result, although not perfect, is good enough, the road area stands out well from the rest of the image.

VII. Conclusions

The objective of this project was to try to predict the 16x16 pixel patches of aerial images. A small amount of training data was given, and pre-processing and data augmentation techniques were used to increase the number of patches to have a dataset big enough to obtain good results.

Different models were tried, mostly making use of Convolutional Neural Networks. The problem with these types of models is that they require a high computational power to give results fast enough, and therefore they could not be optimized to their limit.

The best result was obtained from a CNN that received as input the 16x16 patches of the images with a padding around them to increase the amount of information. The patches were still in RGB format to be able to keep the information from the colour layers, as we thought it was useful due to roads being grey most of the time. The output images then when through an autoencoder, which removed noise from the images, improving the final result.

From this project we have learnt that image segmentation is a difficult process. It requires a high computational capacity to be able to use CNN. For this type of machine learning models, it is more important to have a great amount of data and a bigger model than doing a great amount of pre-processing, as the CNN can include many filters to try to obtain the best possible output.

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015.

[2] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.

[3] Bernd Jahne, Peter Geissler, and Horst Haussecker, editors. *Principles of filter design. In Handbook of Computer Vision and Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 1999.

[4] Irwin Sobel. An isotropic 3 3 image gradient operator. 02 2014.

[5] Andrew Ng, Jiquan Ngiam, Chuan Yu Foo, and Caroline Suen. Zca whitening. http://ufldl.stanford.edu/wiki/index.php/Whitening. Accessed: 2017-12-17.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.