Road Segmentation challenge

NICCOLÒ SACCHI, VALENTIN NIGOLIAN & THEO DEPRELLE
GROUP: RoadSegmentationFault
École polytechnique fédérale de Lausanne

December 21, 2017

***Abstract*—The goal of this project is to segment satellite images in order to discriminate the roads from the background. Several approaches have been tried in the past, most of them requiring a lot of preprocessing and none of them reached great performance. Invented in the late 20th century, convolutional neural networks (CNNs) were only recently successfull in image classification and segmentation problems and performed considerably better than previous techniques. The models we propose in this project consist of two neural networks whose input can be an image of any size and the output is the respective mask of roads. Both models make use of on-the-fly data augmentation, dropout layers and LeakyReLU or ReLU activation functions to reduce overfitting and improve the training. We will also show that, according to our tests, our models outperform the considered baselines.**

I. INTRODUCTION

The segmentation problem can be defined as the matter of assigning a class label to a set of pixels to classify parts of an image. The goal of this project is to segment the roads in satellite images to extract the mask of the roads. The metric used to evaluate the prediction of the roads is the F1 score that combines the precision, i.e. the fraction of correctly predicted roads among all the predicted roads, and the recall, i.e. the fraction of correctly predicted roads among all the true roads. The formula of the F1 score is the following:

$$F1 = 2\frac{pr}{p+r}$$

$$\text{where} \quad p = \frac{TP}{TP+FP}, \quad r = \frac{TP}{TP+FN}$$

Only in recent years, thanks to the increase in computing performance and the possibility to exploit GPUs to parallelize computation, CNNs were able outperform most of the techniques in solving different tasks. The strengths of the CNNs derives from a hierarchy of filters which are learned during the training phase. Using those filters, CNNs are capable of recognizing features of increasing complexity, i.e. higher level representation of the input as the image goes deeper in the network. However, one drawback of CNNs is the need of a large, fully-labeled dataset in order to extract the patterns proper of the objects that have to be recognized, otherwise the CNN would tend to overfit and just memorize the training dataset. The following sections give an overview of the tried techniques focusing in particular on the two neural networks we implemented. Section II gives an overview of the dataset, Section III proposes some approaches to the problem and explains the design of the two implemented networks, Section IV shows and demonstrates the results obtained with our models.

II. EXPLORATORY DATA ANALYSIS

We were given two sets of images acquired from GoogleMaps: (1) a train set composed by RGB images of size $400*400$ pixels and the respective ground-truth images, i.e. images of the same size where white pixels represent roads and black pixels represent the background, (2) a test set composed by RGB images of $608 * 608$ pixels whose ground-truth has to be predicted. The aim of the project is to label the patches of size $16*16$ pixels. By taking a look at the train set and the respective ground-truth we could notice that this task is not trivial. Indeed, the surface of the road is for instance often covered by trees, shadows, buildings, railroads and/or cars, parking lots are sometimes marked as road and the roofs are of the same color of the roads. Those problems can be solved only if we also consider the neighborhood of each patch to correctly classify it. That task can be achieved only with proper feature extraction, either manually, with complex and advanced techniques, or automatically by employing a CNN. Moreover, very few images have peculiar characteristic that differ from the rest of the images. Those are images of highways (3% of the the dataset) and sea sides (2% of the dataset). Some of these complications can be seen in Figure 1.

III. MODELS AND METHODS

The first techniques we studied involved feature extraction using interesting, yet complex and not very efficient techniques that fail in extreme cases, e.g. if there are object obstructing the view of the road or a change of brightness through different images. Such techniques include, for example, computation of the gradient at each pixel (to exploit the fact that roads have continuity in one direction, i.e. a gradient $\sim$0) or the use of an edge detection algorithm to extract the boundaries of the objects and the use of vector

Figure 1. The left image is one of the very few sea sides images, the left image shows that roads may be covered by trees

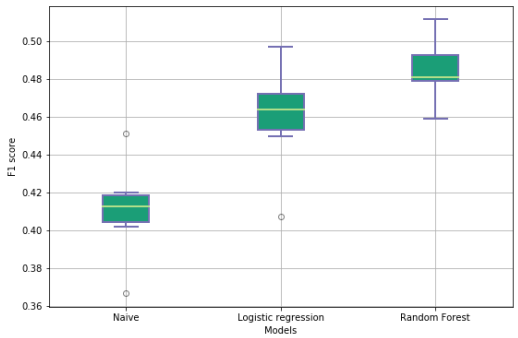

Figure 2. Distribution of the F1 scores computed on a 6-fold cross validation on the three considered baselines.

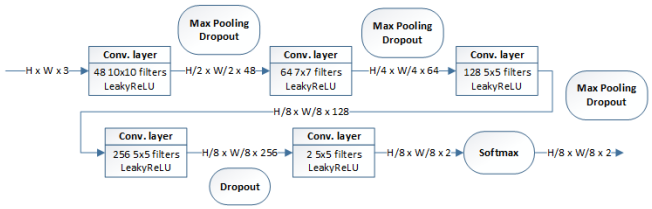

Figure 3. Scheme of the CNN model which obtained an F1 score of XXX

graphics to represent images with geometrical primitives such as points, lines, curves, and shapes or polygons [1]. Given the complexity of those solutions and the poor quality of the results compared to CNNs, we opted for the latter which are well known for the little preprocessing needed and great performance in solving this kind of problem. We start from three very simple baselines: (1) a model that predicts all as road (notice that, since the metric chosen is the F1 score computed on the road predictions, a model that predicts all as background would have higher accuracy but an F1 score of zero), (2-3) we consider each $16 * 16$ image patch as an input and for each one of them we produce 6 simple features extracted from the mean and the standard deviation of each RGB channel within the patch. We use the obtained dataset to train a logistic regression and a random forest. The F1 scores of these three baselines is shown in Figure 2. As expected, this analysis shows that those methods fails in this complex task if more advanced features extraction is not in place. Therefore, we study solutions based on CNNs and we find two main approaches: (1) design a CNN which segment the input image and outputs directly the mask for the roads [2], (2) reduce the segmentation problem to a classification problem, i.e. build a CNN that receives as input a single patch, possibly with a bigger context (neighborhood of that patch), and labels that patch as either road or background [3]. We opt for the first choice as it requires almost no preprocessing. In particular, we implement two types of CNN models, a more traditional one and one based on the U-Net architecture. The two models, which are described in the two following sub-sections, are accompanied by slightly different pre-processing and post-processing processes and data augmentation has been performed differently. Both models can take an image of any size as input and output the respective mask of roads. The training of both the models has been improved by changing the training set: (1) we have removed the seasides images, since these images are not present in the test set, (2) we have replicated the highway images, so to improve the predictions of wide roads. These changes improved the predictions of the test set as will be shown next.

A. *Traditional CNN model*

We designed this CNN in such a way that it reduces each side of the input image by a factor equal to the width of the patch we want to predict in order to obtain an output mask of size $H/n * W/n$ where n is the width of the patch. In particular, we observed better results when predicting patches of size $8*8$. The structure of our CNN is depicted in Figure 3. Since we experienced problems of GPU memory filling (when using a NVIDIA GeForce GTX 960), we could only adopt convolutional layers of higher number of filters in deeper layers, i.e. after reducing the size of the input image. In this way, the first layers have bigger inputs and less filters while the last layers have smaller inputs but can have more

Figure 4. Activations of the first convolutional layer: with ReLU on the left and with Leaky ReLU on the right. As can be noticed the number of dead filter decreased sensibly meaning the model was learning more reasonable features.

filters [1].

The most common activation function used after convolutional layer is the ReLU activation. However, since we have experienced the problem of dead filters [2], we have switched to the Leaky ReLU, a variant of the ReLU which has a small positive gradient for negative inputs:

$$f(x) = max(\alpha x, x)$$

Where α is the slope for negative inputs and is usually a small value ~ 0.01. This choice solved the problem of dead filters as can be seen in Figure 4.

We used k max pooling layers with a pool size of $2 * 2$ where 2^k is the size of the patch we wanted to predict, e.g. to predict $8 * 8$ patch-wise we adopted 3 max pooling layers so to reduce the size of the input image by a factor of 8 on each side. Note that increasing the strides of the convolutional layer would lead to the same size reduction but we opted for max pooling so to automatically select the highest activations.

Moreover, after each *max pooling* layer we added a *dropout* layer, a regularization technique used to prevent overfitting by randomly setting to 0 a fraction of the input during the training phase. We have set the dropping rate for all the dropout layers to 0.25.

Data augmentation: This model works with batches of images generated on-the-fly based on the training set. In particular, the batches are generated by applying rotation and/or mirroring to the each image in order to artificially increase the size of the training set. While each image of the batch is mirrored with a probability of 0.5 we decided to not rotate each image by a random degree since most of the images, both in the test set and train set, have vertical and horizontal roads and we wanted our model to master the recognition of those roads. Rotation is done in the following way: an image of the batch is rotated by any degree with a probability of 0.2 while is rotated by $90° * k$ where $k = 1, 2, 3, 4$ with a probability of 0.8. Figure 5 shows an example of data augmentation applied on an image. This data augmentation proved to be effective and our CNN model learnt not only to recognize horizontal and vertical roads but also diagonal roads.

Post-processing: Once the model have been trained, the predictions can be further improved in order to increase the F1 score. However, even if roads have some intuitive properties, e.g. starting from a road you can always reach the border the image, it is not trivial to improve a predicted mask of roads such that those properties are fulfilled. Therefore, we opted for two easier solutions that proved to increase the F1 score sensibly.

- We compute on the predictions of the training images the optimal threshold, i.e. if the probability of a patch is beyond that threshold then is labeled as road. This step is done with a grid search and by retrieving the threshold which corresponds to the maximum F1 score. This estimated optimal threshold will be used on the predictions of the test set to label each patch.
- To obtain the predictions of a test image, we mirror it and/or rotate it by $90° * k$ where $k = 1, 2, 3, 4$ so to obtain 8 different predictions. Those predictions are then averaged to obtain the final mask of roads.

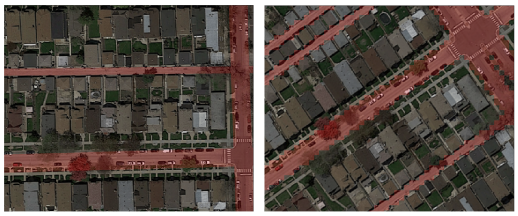

Figure 5. The left image is the original image. The right image is an example of augmentation of that image. After rotating the image, the points outside the boundaries are filled by reflecting the image so to always obtain an image of the same size $(400 * 400)$.

B. U-Net architecture

The second model we have tried is based on the U-Net architecture [**?**]. This network was made to be used for medical image segmentation. Since our problem is close to what this network was made for, UNet was an ideal candidate.

U-Net network architecture is illustrated in 6. It consists of a down-sampling path (left side) and an up-sampling path (right side). Both share the architecture of a traditional CNN and consist of the repeated application of two 3x3 convolutions each followed by a Rectified Linear Unit (ReLU). However, while the down-sampling path contains 2x2 max pooling operations with stride 2, that reduce the dimension of the input, the up-sampling path up-convolutional layer that increase the size of the input back to the size of the input image. Moreover, we are using four skip connections that performs a concatenation with the input of the maxpooling and the output of the up-convolution.

is this sentence correct?

We added to the existing architecture a batch normalization layer after every ReLU layers, a dropout layer before every max pooling and a fifth maxpooling operation.

The energy function is computed by a pixel-wise softmax S over the final feature map combined with the cross entropy loss function E.

$$S(x) = \frac{exp(a_1(x))}{exp(a_1(x)) + exp(a_0(x))} \quad (1)$$

[1] The output of a convolutional layer, when strides is $1 * 1$ and padding is used (which is our case), is $H * W * \#filters$ where $H * W$ is the size of the input.

[2] Dead filters are filters that always output zero for any input. This may happen when, during training, the weights are updated in such a way that the neuron will never activate on any input again which lead to a gradient of zero and no update of the weights.

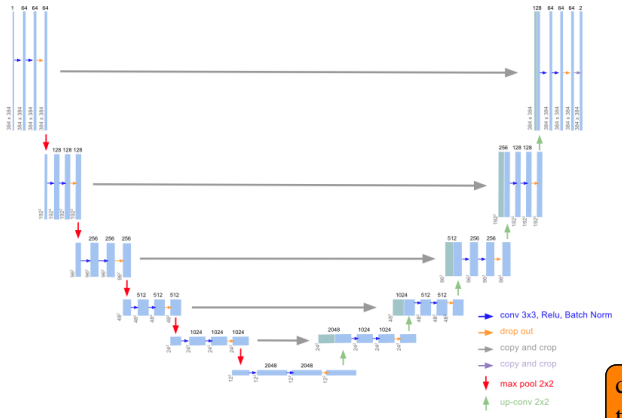

Figure 6. UNet architecture

$$E = \sum_{x \in \Omega} w(x).log(S(x))$$

Where $a_k(x)$ denotes the activation in feature channe at the pixel position x of the image Ω.

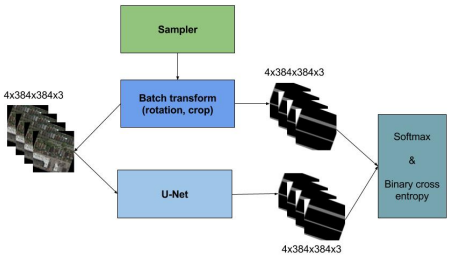

Figure 7. Training step diagram.

Data augmentation: We trained the network batches o images. Each image of the batch is a randomly selected pa of size $384*384$ from the original image. Using this meth our result were good but it appears that diagonal roads w not detected correctly. This was because the majority of images of the training set did not had diagonal lines. correct this problem we randomly rotate the image a every training step. This allows us to have a better diago roads detection since the train set now included roads different orientations, see Figure 8. However, rotating image may leave some pixels outside the boundaries. decided to label those pixels in the ground-truth with different class, i.e. class 2 (ignore part). Those labels allow us to ignore the respective pixel in the computation of the loss, which was properly modified, so that they have no impact on the training process.

IV. Results

A. Traditional CNN

The traditional CNN was trained for XXX epochs, each consisting of XXX batches of 4 images, on a NVIDIA GeForce GTX graphics card with 2 GB of VRAM. In order to compare different CNNs models we used 4-fold cross validation. Figure XX shows the estimated F1 scores of our main CNN models compared to the aforementioned three baselines and agrees with the F1 score obtained on Kaggle (XXX%). Even without any preprocessing other than data augmentation our CNN outperformed the baselines and confirmed their superiority in solving this kind of tasks.

change this, we can say we don't use the cross validation but precision-recall method =¿ add to figure 9 the line representing the traditional cnn

Figure 8. On the left we can see that the diagonal roads are not detected well. On the right, after randomly rotating the images, the model is capable of correcly predict all diagonal roads.

B. U-Net

After 10000 iteration with a learning rate $\lambda = 10^{-3}$ we have obtained a accuracy of 93.35% on the test set where the labels are unknown. To compare different techniques we used the Precision Recall method (PR) which focuses on the relevance of the prediction on the labels of a test set. To compute the PR values we split the training dataset into two subsets, one used for training and one used to compute the PR curve. The latter is computed by gradually increasing the threshold (used to establish, given its probability, if a pixel is a road) to compute different PR values. In this way, we obtain the graph in Figure 9. We can observe in this

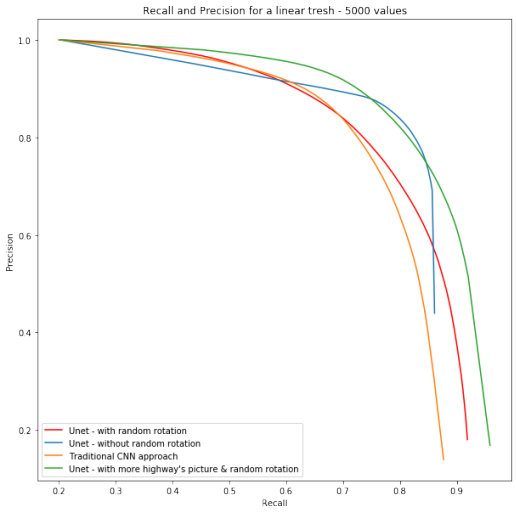

Figure 9. Precision and recall evolution regarding our different methods

plot three important things. Since the yellow curve is bellow every others, we can say that Unet is perfroming better than

the traditional approahch. The second observation is that the green curve is above the red one. Thanks to this we can conclude that adding more hightway images and getting ride off the sea sides images was a good thing. The third observation is about the blue curve. We can see that in some cases the blue curve is above the green one. In fact when we are not rotating the image the network learn better how to detect verticals and horizontals lines but for the others the result is worst. That explains why the blue curve goes above the yellow one between 0.6 and 0.8 but overall the yellow is the one with the better results.

V. Conclusion

Both models proved to be versatile and effective in predicting correctly. However, our traditional CNN model failed in recognizing narrow roads, especially when they were covered by shadows, those types of roads are indeed almost inexistent in the training set while present in many images of the test set. Moreover, the train and test datasets were mainly restricted to cities. To obtain a model that would efficiently work with a wider range of input images, e.g. images with snow, country images or on different meteorological conditions, we would require a bigger and diverse training set.

References

[1] J. Hormese and C. Saravanan, "Automated road extraction from high resolution satellite images," *Procedia Technology*, vol. 24, no. Supplement C, pp. 1460 – 1467, 2016, international Conference on Emerging Trends in Engineering, Science and Technology (ICETEST - 2015). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2212017316302705

[2] K. Lis and I. Roesch, "Road extraction from aerial images," 2016.

[3] D. Pavllo, M. Martinelliy, and C. Hwang, "Epfl machine learning project 2 report road segmentation," 2016, https://github.com/dariopavllo/road-segmentationl.