Machine Learning CS-433 - Class Project 2 - Road Segmentation

Lucas Braz (343141) – Clement Nicolle (344576) – Pierre-Alain Durand (344313)
Department of Computer Science, EPFL, Switzerland

***Abstract*—In this report, we explain our U-Net implementation trained to classify roads on Earth satellite images. We used convolutional layers with padding and 2D Max Pooling in the encoder and transposed convolution in the decoder part, each with 2D batch normalization and ReLU activations. A key improvement was the data augmentation because our training dataset was only composed of 100 images. We flipped images and did some rotations of 45° or 90°. We achieved a 88.4 F1 Score on the AIcrowd website dataset test after only one hour of training for 35 epochs with our recent GPU RTX 2060 laptop.**

I. Introduction

In this project we had to perform what is called a semantic segmentation of roads on a set of satellite images from Google Maps. We were given for that, in addition to the 50 RGB satellite images of size 608x608 to categorize for the challenge, a set of 100 RGB satellite images of size 400*400 with their associated groundtruth (binary) where each pixel is labeled either as road (white) or background (black).
We had to design a classifier to differentiate roads from other objects in these images. We therefore realized a derivative of a Deep Learning model specialized in semantic segmentation, the U-Net [Ronneberger et al. (2015)]. To train it, we performed a moment SGD and we decreased the learning rate if the F1 score of validation went down a few times in a row. For submission, our predictions are evaluated at the level of a 16×16 patch, and not pixel by pixel. We will detail this in this paper. In section II, we present the models we have developed to solve this problem. Section III talks about data augmentation, which is essential for this task given the few images initially given. Section IV briefly defines the libraries and software tools we used for the implementation and shows some intermediate results of our model computation. Section V introduces some automatic post-processing we tried on the model outputs in order to improve results. Finally, Section VI gives an overview of our results. We end with a conclusion and discussion of the areas of improvement.

II. Unet

The U-net (Ronneberger et al., 2015) is based on the concept of deep neural networks of the encoder-decoder type. It is entirely composed of convolutional layers and uses Maxpools to gradually reduce the information in the encoder and transposed convolutional layers with stride in the decoder. This compression leads to an informational bottleneck in the middle of the network implying a need for the model, during the learning process, to select relevant features at each step in order to obtain good results Here is its graph :

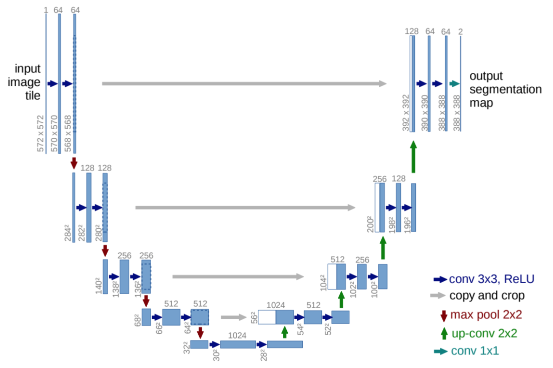

Figure 1. U-Net Vanilla Deep Learning Network

When convolving without padding, which is used in the Unet Vanilla, the image size is modified by each 3x3 convolution, we lose one pixel on each edge. Although we have also implemented this Unet Vanilla, we have chosen to realize 3 x 3 convolutions with zero-padding (or réplication padding) in order to remove the need for a crop when copying images from the encoder to the decoder and to obtain at the output of the network an image of identical size to the input (and thus to the labels) without additional modification.

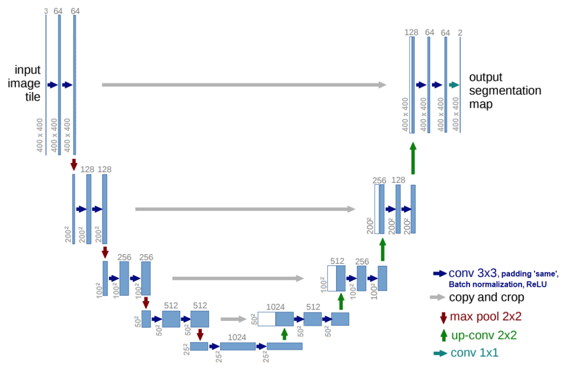

Figure 2. Our U-Net Implementation

In input we have images of 3x400x400 in the training set and 3x608x608 for the test images. In output we obtain an image of dimension 1x400x400x2 (resp. 1x608x608x2 for the test set). The last 2 dimensions correspond to the binary classification of the image. We apply a LogSoftmax on the output (on Pytorch through the Cross-Entropy criterion function) and thus obtain for each pixel the entropy (think "logarithm of probability") associated to the fact that this one is a background (first dimension) or a road (second dimension).
Our Loss was a Negative Log Likelihood (on Pytorch through the Cross-Entropy criterion function, obtained by combining LogSoftMax and Negative Log Likelihood). We also tried the implementation of the Jaccard criterion (or IoU, area of overlap divided by area of Union), which gave satisfactory results.
Two optimizers were implemented, the first one was a SGD with momentum (with 0.9 of momentum, chosen arbitrarily, allowing to give inertia by letting the previous gradient have an impact on the current gradient calculation) and the second one was the Adam optimizer [Kingma and Ba (2014)]. We found that the Adam optimizer was not more efficient than the SGD, so we chose to keep the simple SGD with momentum optimizer. We also implemented a Learning Rate Scheduler allowing an exponential decay (gamma = 0.95). The learning rate was thus lowered if the validation F1 was lowered 5 iterations in a row.
We tried the dropout, without convincing results. On the contrary, the batch normalization showed excellent results, which is why we constantly implemented it afterwards.

III. Data augmentation

A convolution operation allows to retrieve a type of information on an image by translating a filter on it. There is thus invariance in translation but the operation remains sensitive to the rotation. A quick analysis of the dataset showed that there was a serious lack of images with diagonal roads compared to the test set. We therefore increased our dataset of rotated images by 45°. Moreover, we quickly noticed that with only the 100 basic images, the network quickly obtained an F1 of 96% on the training set and only 80% on the valid set. We were cruelly short of images. We then significantly enriched our dataset by performing many 90° rotations and different flips of our images. With 800 images (including the 45° rotations), we already had a better ratio between training and validation.

A. 45° Rotation

The use of the 45° rotation of the images was necessary in order to enrich our database, in particular by bringing many diagonal roads as we just explained. As it is illustrated on the figure [3] below, by rotating the image in this way, edges appear at the 4 corners of the image in addition to losing the initial corners of the image which go out of the 400x400 frame.

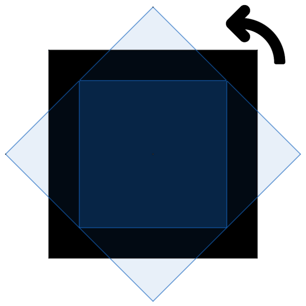

Figure 3. Illustration of the need to crop the image after a 45° rotation

To get rid of these edges we must then reduce the size of the image. By applying the Pythagorean theorem we know that we must extract the center of the image of a size 283 x 283. For reasons of compatibility with our model which includes 4 poolings 2x2, dividing by 16 the size of the image, we wanted ideally that the width of the image is a multiple of 16 less than or equal to 283. We have therefore crop the center of the image with a size of 272x272.

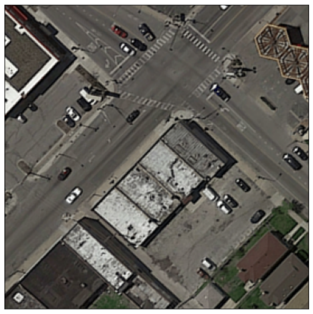

Figure 4. Image rotated with diagonal roads

IV. Python librairies and software implementation

For implementation, we used Pytorch [Paszke et al. (2019)]. This python library allows to create the graph of our model and to keep the associated weights, connections and gradients during a gradient descent. We define an optimizer, a Loss criterion and a scheduler for the learning rate and we can easily perform forward and gradient descent then test our model on our validation set. Finally, we use our model on test set and create the submission file with it's output. For visualisation we simply used matplotlib.pyplot.
We also used sklearn to split training and validation set, associated with numpy. We used scipy and OpenCV2 for some pre-processing and post-processing. Finally, we used tqdm to print a beautiful progress bar during training.
We trained locally on one of our laptop with a recent GPU

RTX 2060 laptop and 8Gio RAM. We also used Google Colab and it was, in fact, faster than our computer.

V. Results

Name	p_{drop}	45°	flip	Optim	f_1
U-Net Vanilla	0.0	X	X	SGD	0.72
U-Net 2	0.2	X	X	SGD	0.71
U-Net 3	0.2	✓	✓	SGD	0.78
U-Net 4	0.0	✓	✓	Adam	0.8
Our best U-Net	0.0	✓	✓	SGD	0.884

Table I
$\mathbf{p_{drop}}$: DROPOUT PROBABILITY, $\mathbf{45^\circ}$: SIMPLE ROTATION, **FLIP** : SIMPLE FLIP, **OPTIM** : OPTIMIZER USED, $\mathbf{f_1}$: TEST-SET f_1 SCORE ON AICROWD

After training, convolutions extracts meaningful features from images, we can then observe one output of each layer from the image to the prediction as in the figure [5].

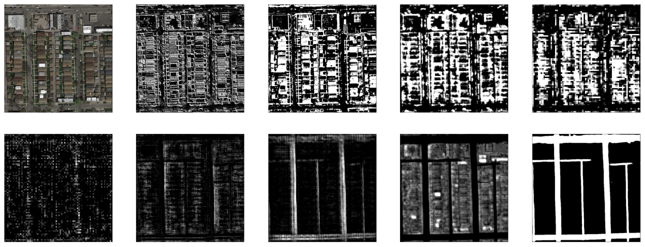

Figure 5. Differents steps in the model convolutional layers

We can clearly see with the figure [5], first the loss of information in the images during the encoder due to the different poolings. Then, during the decoder part, we can see the clear distinction between the roads and the background appear progressively.

We can analyse some predictions on our validation set compared to their targets, as in figure [6] :

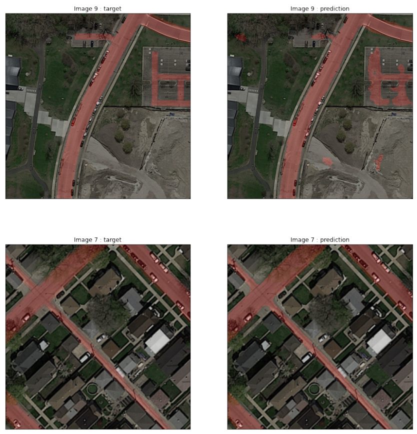

Figure 6. Target overlayed on validation images on left, model prediction overlayed on the same image on right

We can see that, on validation test, it's quite perfect, unless some complicated parts where the road is only moderately defined on land. We can then see some output for our testset that we have submitted, on figure [7] :

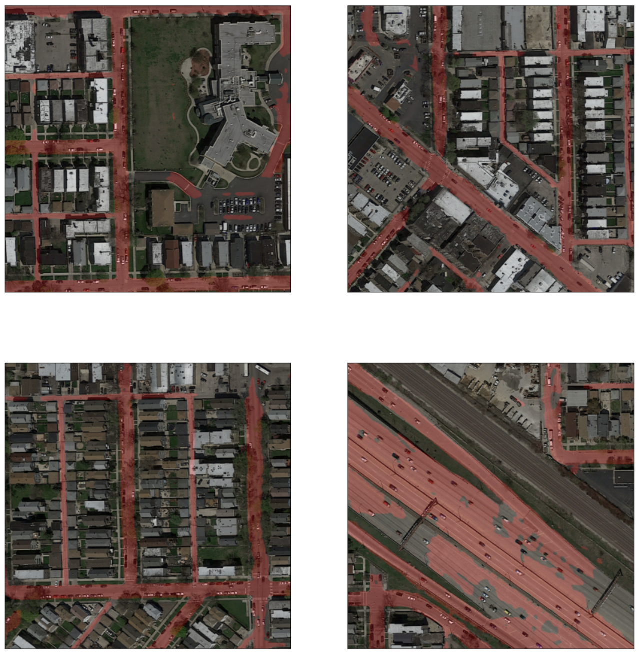

Figure 7. Selection of submitted pictures with road prediction

Results are good overall but we can notice some recurrent errors. Indeed, some roads are often interrupted, we note this phenomenon on the small roads especially because it is more exposed to drawbacks such as the shadow of buildings or trees that come to cause cuts.
Moreover, the large highways visible on the image at the bottom right is difficult to determine with the presence of many vehicles, billboards coming to cut the road and especially its unusually large size for our model.
Nevertheless, we can clearly see that the diagonal roads are now as successful as the horizontal roads. The 45° rotations as data augmentation are an essential element in our algorithm to better predict the whole of our roads.
To improve the continuity of the predicted routes and avoid the few isolated spots predicted as routes. We planned a post-processing based on morphological operators of binary opening and closing.

VI. Discussion

At the beginning we were stagnating around 80% of F1. We chose to resize the test dataset to 400x400 before running it in the model and to put the output in 608x608. By stopping this useless approach our results are instantly back. Indeed, reducing the size of an image in this way causes the loss of a lot of information, a loss of precision which led to a

decrease in our final score.
We also tried a post-processing based on mathematical morphology operators. We wanted to remove areas where small spots appear when they should not and also fill in holes on the predicted routes.
We proceeded in two steps. First of all, we made an opening with a 5 x 5 square as a structuring element, which allows us to remove all the spots that are not at least this size. Then in a second step, we realize a closure with as structuring element a line of size 1 x 10 horizontal, vertical or diagonal which allows to join holes on the roads or between predicted roads.
The structuring element "line" created problems, however, because it joined roads in the direction of the line that met along another axis. We then tried to simply make the opening but this unfortunately removed true positives in addition to false positives when these true positives were also isolated, so our F1 was slightly lower. So we stopped experimenting with this post-processing, even if with more time it could have eventually improved the results. The guideline (Hough image processing algorithm for recognition of the main guidelines of an image) of each route should be taken into account and a different post-processing should be applied to each road to optimize it.

References

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *CoRR*.