Machine Learning Project 2: Road segmentation challenge

Baldwin Nicolas, Leidi Mauro, Roust Michael
Department of Data Science, EPFL Lausanne, Switzerland

I. Abstract

The Aim of project is to build a model able to segment satellite images into roads and non-roads zones. The model can be trained on the EPFL road segmentation challenge dataset only. We will show how U-Nets models are able to achieve high accuracy on this task. By performing data augmentation and an auxiliary loss technique we achieved a F1-score of 90.7% and an accuracy of 95%.

II. Introduction

The dataset of this challenge is composed of 100 RGB satellite images of 400x400 pixels together with their masks which are gray-scale images of the same size. The test set is composed of RGB images of 608x608 pixels. The goal of this project is to build a model able to segment satellite images into two classes: road (=white) and background(=black). The mask are gray-scale images containing all sort of grays, not only white and/or black pixels. In order to find the final F1-score the images are separated in 16x16 pixels sub-images. For each sub-image if the average of the pixels values exceeds 0,25 than the sub-image is mapped to white, otherwise it's mapped to black. The resulting image is 25x25 pixels and 38x38 pixels for the train images and test images respectively and contains only black and/or white pixels. The same procedure is applied to the mask and then the two resulting images, that are now only composed of black and white pixels, are compared to compute the performance metrics. To face this challenge we decided to use a convolutional neural network, more specifically a U-Net which has proved to be one of the most suitable architectures for image segmentation on small datasets [1].

III. Models And Methods

A. Exploratory Data Analysis

The most important concepts to keep in mind about the data are the following:

- As shown in the images presented in Figures 1 & 2, our model should be able to recognise the roads even if trees or other objects are present, and should also be able to distinguish between roads and parkings.
- In Figure 3 we report the histogram of the distribution of the pixel values within the training data masks. We can see an imbalance between the two classes, many more black pixels than white are present. The average value is 0.1868. We will have to choose an appropriate loss function so that the model does not tend to predict black too often.
- In Figure 3 we can also notice that many gray values are present. By studying the values present in the images we realize that gray pixels are the contours of the roads. An example is presented in Figure 4. We believe that the masks were labeled by hand by a group of people and, subsequently, the resulting masks have been aggregated somehow, and for this reason values other than 0 or 1 appear.

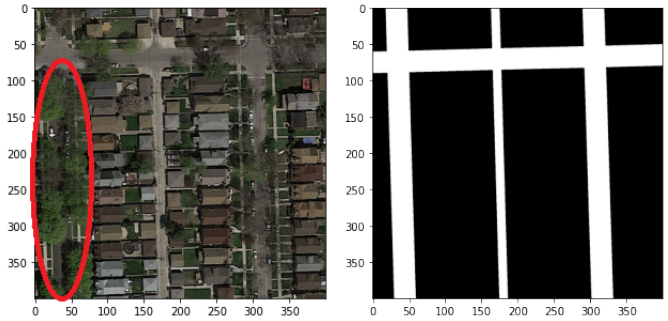

Fig. 1: The model should understand there is a road behind trees.

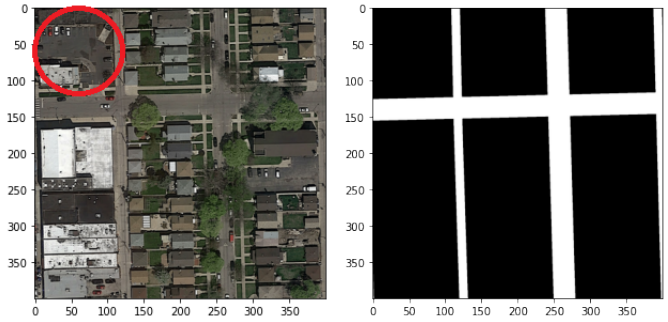

Fig. 2: The model should understand that a parking is not a road.

B. Data Augmentation

The training set is only composed of 100 images and their associated masks. In order to increase the robustness and performance of our model [2], and reduce the risk of overfitting, we performed data augmentation. For every image present in the training set we performed:

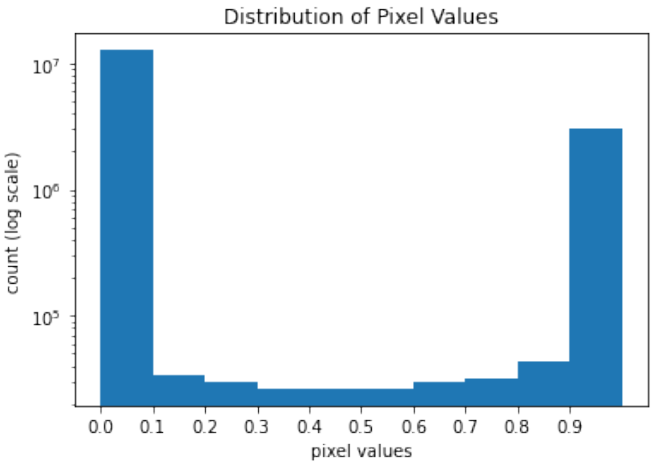

Fig. 3: The distribution of the pixel values in the dataset masks. We can observe that the background cover a bigger surface than the roads and that the picture is grayscale instead of only black and white.

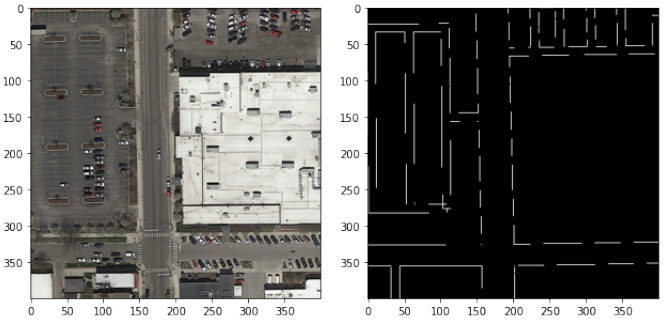

Fig. 4: We observe that the values different from 0 or 1 are the edges of the roads.

- Rotations: for every image we rotate it with ten random angles and crop them back to the original size (400x400) pixels. These transformations are done on both the images and on the masks. This process generate 1000 additional images.
- Mirroring: we flip every image horizontally and vertically. This transformations are done both on the images and on the masks. This process generate 200 additional images.
- Color changes: we transform to gray-scale and create a color jittered version of the picture, in this case the transformation is done only on the image and not on the mask. This process generate 200 additional images.
- Random combination of all the previous transformation (10x). This process generate 1000 additional images.

Note that once the data is augmented, it is impossible to fit all the images in the RAM, therefore it is necessary to implement a dataloader that stores only the filepaths in memory and load images only when it is needed.

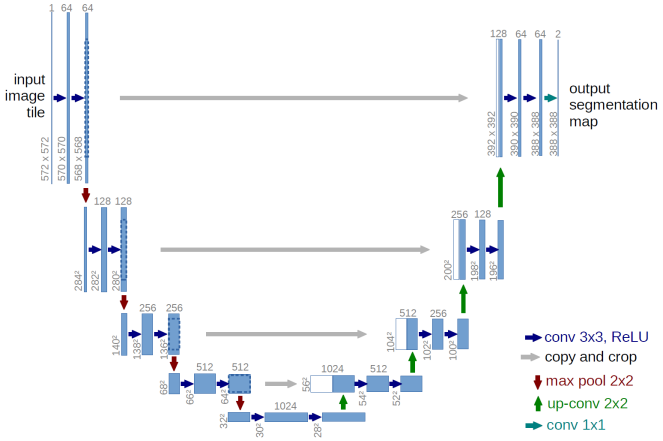

Fig. 5: The architecture of the original U-net presented in the paper [1]

C. Learning rate gradual decrease

To train our U-nets we perform a stochastic gradient descent using adam optimizer. Furthermore, using the validation set, we monitor the F1-score that the model achieves in the current epoch. Each time the F1-score does not exceed the previous best F1-score by at least 0.001 for 4 consecutive epochs, the learning rate is halved.

D. Baseline Model

To start tackling this problem we decided to use a U-Net, proposed by Olaf Ronneberger, Philipp Fisher and Tomas Brox in Convolutional Neural Nets for Biomedical Image Segmentation [1]. We choosed to start from this model because it performed well on small datasets. The U-Net is a fully convolutional neural network, where the contracting network is supplemented by successive layers, where pooling operations are replaced by upsampling operators. The model architecture is illustrated in Figure 5: its first part is composed by four separate different max-pooling (2x2) operations, each-one is alternated by two consecutive convolutions (3x3) with ReLu activation. After each pooling operation the number of feature channels is doubled. Subsequently, the second part of the model is composed of a mirror architecture in which pooling operations are replaced by upscaling operations. Please see [1] for more details. We downloaded the pre-trained model from pytorch [3] and proceeded to further train it on our training set. The performance of this model was very bad as it only predicted completely black images (= no roads). This is due to the imbalance between the two classes. (As a loss metric at first we chose cross-entropy loss)

E. First Model

To solve the baseline model problem we created a new model by changing the loss function. The loss function chosen is the BCEWithLogitsLoss which combines a Sigmoid layer and the BCELoss in one single class, and allows you to manage the unbalanced classes. This model achieves much better performance than the previous one. To deal with the

unbalanced data, we provide to the BCEWithLogitsLoss function a positive weight matrix (which is the size of the image at the output of the model). Each pixel $p_{ij}^{pos-weight}$ (pixel of column i and row j) of the positive weight matrix is calculated with the train set and is given by:

$$p_{ij}^{pos-weight} = \frac{nb - images - with - p_{ij}^{image} - white}{nb - images - with - p_{ij}^{image} - black}$$

F. Second Model

To further improve performance, we augmented the dataset as mentioned in section B and trained a model with the same architecture.

G. Third Model

For the development of the third model we followed an intuition. The goal of this challenge is to obtain the best possible F1-score, by performing the following procedure: taking average on patches of (16x16) pixels and thresholding them at 0.25, then comparing the result with the image obtained by performing the same operation to the mask (as described in the introduction section). However, although it is true that, if our images are identical to the masks, then the F1-score will be 1, we can try to help the model understand that it must also have a coherence on the averages of the patches (16x16). To do that we have to act on the loss function. Instead of training our model to learn only masks, we decided to add a last pooling layer to the model output which does average pooling on patches of 16x16 pixels. An activation function:

$$f(x) = \frac{tanh(x - 0.25) + 1}{2}$$

is applied to the output of this layer. We used this function to try to mimic the step function around 0.25 but still maintaining the differentiability necessary for back-propagation. Instead of using a single loss function that pushes the model to learn the mask, we use an auxiliary loss on this last layer, which is compared to the mask, that is average pooled and passed through:

$$f(x) = step(x - 0.25)$$

All this is summarized in Figure 6. The loss function used are L1-Loss for the loss1 and BCEWithLogitsLoss for loss2. The global loss is given by:

$$Loss_{tot} = \alpha * Loss1 + \beta * Loss2$$

We restricted our study to β = 1 - α in order to lower the number of hyperparameters. One last thing that is interesting is the dynamism of our new loss function. Initially loss1 (l1) is dominant, and the model will learn to mimic masks by trying to reduce the absolute difference pixel by pixel. Once this difference is small, loss2 will increase its relative importance, and the model will learn to mimic averages on patches (16x16) rather than focusing pixel by pixel. In this way the model will have an attention both on the single pixel values and on the averages of the patches (16x16).

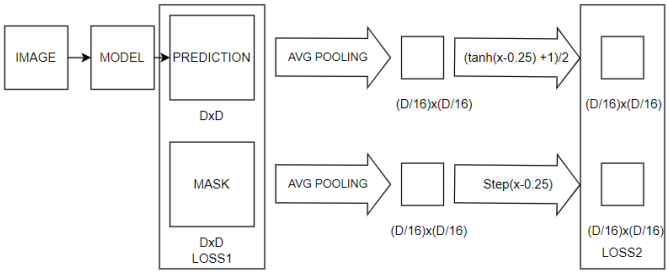

Fig. 6: A diagram representation of our auxiliary loss used in the third and following models.

H. Fourth Model

The last change we want to make to our model is to try to optimize it as much as possible. Having many hyperparameters and realizing that even with a Colab pro account the training time of a model exceeds 30 minutes, cross-validating on many combinations of parameters becomes infeasible. Therefore we identified the parameters that we believe had the greatest impact,α and β, and performed hyperparameter tunning only on these, leaving others hyperparameters, as the dimensions of the model's layers, identical to the original U-Net ones.

I. Validation method

Having a non negligible model training time, we decided to divide the data into train and test, instead of doing cross validation to have a validation metric. In this way we were able to advance much faster in the development of our model. Having a small dataset, coupled with the fact that we applied data augmentation, prompted us to [4]:

- 1) split the data into train and test sets before data augmentation. In this way our metric validation is not biased by the fact that the test set can contain images very similar to the train set (eg.: same image rotated by 1 degree).
- 2) Perform data augmentation on the train set but not the test set. Validate the results on trained models only on 80% of the data. Even if the reached accuracies are lower we are able to compare our different models with an unbiased estimator.
- 3) In the end, our best performing model was trained on data augmented before the split, in order to train it on all data. In this last training our validation metric is biased, therefore the F1-score on AI-crowd will be lower, but we don't care, because the validation done in advance assures us that this is our best model.

Additionally, all models are trained with a batch size of 10, a scheduler as mentioned in C and 30 epochs This procedure was essential to allow us to compare the models, without using AI-crowd each time, which would have led us in some way to use the test data to validate our model. Furthermore, thanks to this procedure, the difference between the AI-crowd scores and those of our validation has drastically collapsed.

The results of cross validation can be seen in table I and table II.

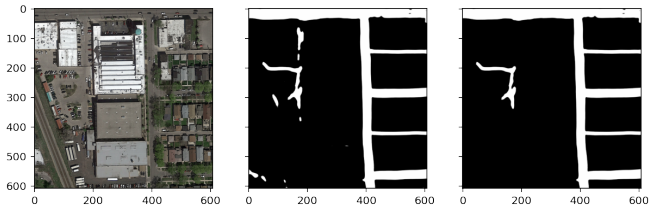

Fig. 7: **Right**: original image **Middle**: prediction made by our model. **Left**: prediction with small connected components removed.

J. Post-processing

In order to further improve our results we attempted to implement some post-processing steps.

First of all, we noticed that our model sometimes labeled small blobs as roads, blobs that we are often too small to be of any significance. See Figure 7 for an example (Middle image).

These blobs often are ambiguous patches of asphalt such are parkings or connections to such parkings and are much more common on our predictions on the test set. Without having access to the ground truth it is difficult even for us to manually determine how these patches of asphalt should have been labeled. So, we found it tolerable that our model couldn't identify such roads well.

Our approach to improve this was to find all connected components on and image and find their area. If the area is below a given threshold we deleted that blob. We manually chose the threshold which we found to signify the minimum area that a meaningful piece of road could have. The rightmost picture in Figure 7 shows the result of this process.

Unfortunately, possibly due to the rounding out of the final submission we didn't manage to improve the score of our prediction with several different area thresholds. Therefore, we didn't implement this post processing in our final pipeline.

We also considered using morphological transformations such as opening/closing and erosion/dilation to improve predictions but the idea was discarded since it was too case-dependent.

K. Final-model

After having found that our best model is model 4 with alpha = 0.5 and betha = 0.5. We retrained the model for 100 epochs on 2000 augmented images with a batch size of 10 and a scheduler as mentioned in C. This model achieves a 0.907 F1-score on AI-CROWD.

IV. Results

A. Validation Results & Results On AI-Crowd

As already anticipated, the Baseline model, which incorporates the original architecture does not allow satisfactory results to be achieved. The training converges to a model that predicts a single class, the background. Model1 is the first model that performs satisfactorily. In fact, by changing the loss with the BCEWithLogitloss, the F1-score rises to 88.05%.

Model Name	F1-score (Validation)
Baseline	/
Model 1	0.8805
Model 2	0.9064
Model 3	0.8992
Model 4	0.9084

TABLE I: Results: Validation & AI-Crowd F1-score.
Please note that this are the results coming from our validation method described in the section Validation method, and therefore are the results obtained by training only on 80% of the data, therefore this results are non-optimal for the models, but allows for a non-biased cross-comparison between their performances.

Model Name	hyperparameter	F1-score
Model3	α=0.25 , β=0.75	F1 = 0.8992
Model3	α=0.75 , β=0.25	F1 = 0.9084
Model3	α=1 , β=1	F1 = 0.9031
Model3(=4)	α=0.5 , β=0.5	F1 =0.9124
Model3	α=0.25 , β=0.25	F1 = 0.9066
Model3	α=0.6 , β=0.4	F1 = 0.8610

TABLE II: Results of Validation: hyperparameters combinations and F1-score associated

Model2 shows the positive influence of data augmentation that brings the F1-score to 90.64%. Model3 demonstrates how the auxiliary loss technique does not assure better results in fact the F1-score drops to 89.92% . Finally, in table II, we see that the result of the model using auxiliary loss is very dependent on some hyperparameters, which if fine tuned, allow to create our best model: model4. In the figure 8 the learning curves of the various models can be observed.

V. Conclusion

In this work we have demonstrated the qualities that U-net are able to achieve for image segmentation even with small datasets. We have also seen how the use of auxiliary loss, if implemented correctly, can improve the performance of a model. The limiting factor that accompanied us during development is the ratio between time available and training time of a model. We are convinced that with more time available, even without changing the architecture of our model, better performance can be achieved by tuning other parameters. We also would have liked to have had the opportunity to delve deeper into post processing. Despite this, we are very satisfied with the results obtained.

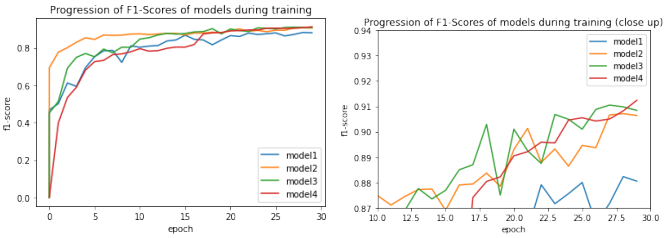

Fig. 8: Training curves of our four models.

External libraries

- PyTorch : Machine learning models. [3]
- Pillow : Basic image processing. [5]
- OpenCV : Advanced image processing (connected components). [6]

References

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". en. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Vol. 9351. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 9783319245737 9783319245744. DOI: 10.1007/978-3-319-24574-4_28. URL: http://link.springer.com/10.1007/978-3-319-24574-4_28 (visited on 12/21/2021).

[2] *(PDF) A Comparison of Data Augmentation Techniques in Training Deep Neural Networks for Satellite Image Classification*. en. URL: https://www.researchgate.net/publication/340294990_A_Comparison_of_Data_Augmentation_Techniques_in_Training_Deep_Neural_Networks_for_Satellite_Image_Classification (visited on 12/22/2021).

[3] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[4] SNOW.DOG. *Data augmentation techniques and pitfalls for small datasets*. en. URL: https://snow.dog/blog/data-augmentation-for-small-datasets (visited on 12/21/2021).

[5] Alex Clark. *Pillow (PIL Fork) Documentation*. 2015. URL: https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf.

[6] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).