Machine Learning Project 2 : Road Segmentation

Martin Cibils, Maxime Fellrath, Yura Tak,
Department of Computer Science, EPFL, Switzerland

***Abstract*—Our aim is to perform road segmentation on satellite images. The task of road segmentation consists of classifying each pixel of an image to road or background class. For this purpose, U-net model is known to be the best. Ours achieved 0.874 of F1-Score.**

I. Introduction

Image segmentation is a very popular problem in computer vision. It can be separated into two problem instances; semantic and instance segmentation, where semantic segmentation refers to the process of linking each pixel of an image to a class label while the other type partitions images into individual objects. In the context of this road segmentation Machine Learning challenge, the aim was to assign pixels in a binary fashion to either a 'road' class or a 'background' in satellite images provided by Google Maps. Among different methods of segmentation (see II), we decided to take the approach of deep learning, more precisely with a U-Net model.

II. State of the art

A few years ago, before deep learning techniques such as convolutional neural networks became the favorite approach to address image segmentation problems, other methods were favored. These included methods like K-means clustering [1] and computer vision based techniques such as thresholding, edge detection, region based and feature based clustering [2]. However, with the emergence of deep learning, the image segmentation field gained remarkable performance improvements with the apparition of different models such as well known convolutional neural networks like simple CNNs, AlexNet, U-Net, etc [3]. As sus-mentioned, for this binary segmentation challenged, we decided to choose a U-Net encoder-decoder. The U-Net was initially designed to work on medical images [4] but is now widely used for many other purposes, because of their incredible performance.

III. Data

This challenge's data set consisted of a pair of a hundred satellite images and their corresponding ground-truth images. The training samples were of size 400x400 pixels and a test set of 50 test images of size 608x608 was also provided.

A. Data Augmentation

For a model that has such a high number of parameters as U-Net, 100 images as training were too few and risked to cause overfitting. To avoid this issue and teach the model invariance properties, we needed more data. Data augmentation was done by performing rotations and symmetries of the original images. We initially tested to implement augmentation using the scipy library, which generated around a thousand more new samples. After seeing the benefits of this on our model's results, we switched to use the ImageDataGenerator class, which generates batches of tensor image data. We used the option fill = 'reflect' which helped avoid losing data in the image angles when rotating the image with an angle that was not a multiple of 90 degrees. An example of the effect of image augmentation can be seen in Figure 1.First row are the original image and ground-truth, and below are example of our data generator. We can observe the reflect filling by looking at the edges of the images, where the road is mirrored into a triangle like shape. For our best model, we added this way 40 images for every one of the 100 given which gave a total data-set of 4000 images.

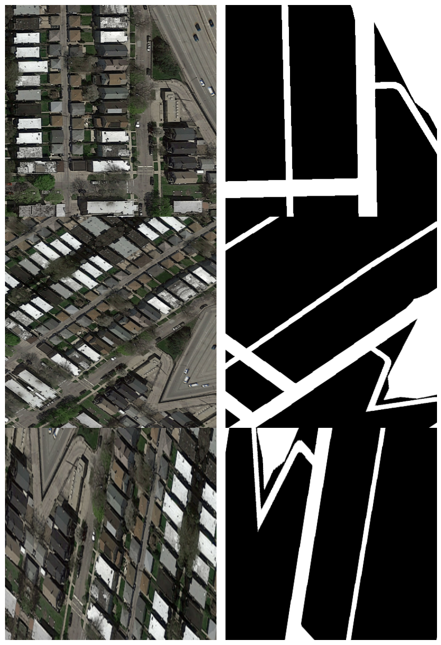

Figure 1. Data augmentation on original image and its ground-truth (upper row).

B. *Simple methods of segmentation*

Before training different neural networks, segmentation was performed using available pre-trained tools. The library pixellib provides a simple interface to segment the image with two different tools respectively for semantic and instance segmentation. Naturally the semantic segmentation was applied on the images and results can be seen in Figure 2. For semantic segmentation, the library uses an Xception model trained on the pascalvoc dataset. As we can see, the result is far away from the ground-truth. The result classified the ground-truth road pixels as road correctly, but also a lot of other pixels belonging to the background. It shows us that the pascalvoc dataset is not enough to have accurate results.

Figure 2. Results of simple segmentation methods as mentioned in III-B. Input image (left), segmented image (middle) and ground truth (right)

IV. Models and Methods

Together with this project, a baseline CNN was provided and we used it as a starting point. Among different neural network models, we principally tested the U-Net.

A. *Baseline CNN*

This model consists of a CNN with 2 convolutional and pooling layers with soft-max loss. This model was trained with an SGD optimizer with momentum and with a learning rate of 0.01 decaying exponentially with the training step.
As the submission is based on 16x16 pixels patches, we fed our network a list of patches of the input images. This had the advantage of being less resource-intensive for the training part of the network. This pre-processing was used only for the baseline.

B. *U-Net*

The U-Net is the state-of-the-art model of this segmentation challenge. It has an encoder-decoder convolutional neural network architecture where the encoder part is the convolutional part, extracting high-level features from the input image. The decoder, on the other hand, is the deconvolutional part, performing our task, the segmentation based on the extracted features. In Figure 3, we can see that there are three direct connections (skip connections) between the layers of encoder and decoder. The skip connections provide spatial information to the decoder to help to reconstruct the image. The output of the U-Net is the probability of each pixel belonging to a road [5] [6].

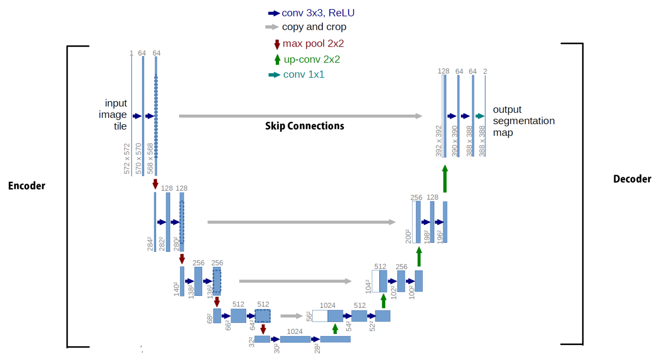

Figure 3. U-Net model

C. *Implementation*

To implement the U-Net, the Keras module of tensorflow was used respecting the architecture of the network with the exception of the first and final layer, as the shapes did not match.

D. *Training*

- Loss: three different types were tested; binary cross-entropy (BCE), binary cross-entropy with a Dice loss [7] and binary cross-entropy with Jaccard loss [8].
- Optimization: both optimizers Adam and SGD with momentum were tested but eventually Adam was chosen, as it fitted the model and its large number of parameters.
- Learning rate: for both optimizers, learning rates of 0.01 and 0.001 were tested but 0.01 was better suited for SGD optimizer while 0.001 for Adam.
- Number of epochs: 100 epochs were used for final training. For fewer epochs, the network already converged, but 100 epochs gave a higher accuracy.

E. *Evaluation Metric*

The model was evaluated using a F1-score and accuracy.

$$F_1 = \frac{tp}{tp + \frac{1}{2}(fp + fn)} \tag{1}$$

Due to the nature of the road images, the data is unbalanced, with more background pixels than road pixels and thus F1-score is more representative than accuracy.

F. *Post-processing methods*

Once our model was ready to use there was still a few operations to make before having our finals results. Indeed, the test image (400, 400) did not have the same size as the test images (608, 608). Our first idea was to use the resize function from skimage. This function worked pretty well but we also tested another method where we found an existing function to create a smooth prediction [9] using image patches to create a bigger image. We used it here only to resize our image but had mixed results. For example it worked pretty good in some case but would also add some random noise in some places (see Figure 4,5).

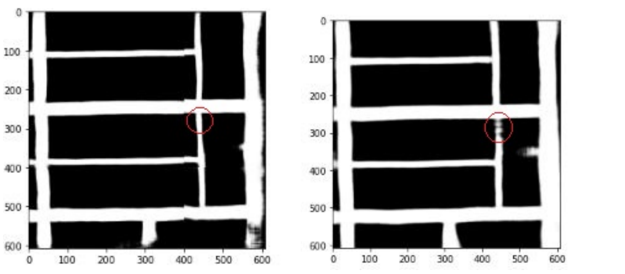
Figure 4. With (left) and without the function as in [9] (right) with good results

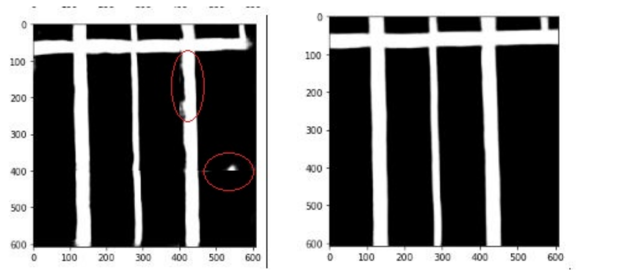
Figure 5. With (left) and without the function as in [9] (right) but it adds noise

In the end, the resize function gave a slightly better result on AIcrowd so we kept it. We also had to find a good threshold value to define to which class belonged every pixel. By plotting a few values it resulted that the optimal value was around 0.5. To conclude this, we computed the absolute loss between the prediction and the given ground-truth on 200 images and kept the threshold with the smallest loss. We found a threshold of 0.504 (see Figure 6). However after testing some submissions, we realized that a threshold of 0.49 achieved a better score on AIcrowd. This could be due to the fact that we are trying to find a good threshold on the training set but the final result is computed on the test set.

G. Results

As explained before, we evaluated our results according to the F1-score. We tested on the 50 images of the test set and the results can be seen in Table IV-G. As expected, the U-Net performs very efficiently and best using a Binary cross-Entropy loss, with a F1-score of 0.874 on AIcrowd for our best model.

V. Discussion and Further Ideas

As expected, we easily came with a solution that obtained much better results than the baseline did, which made sense when looking at the simplicity of the model used in the

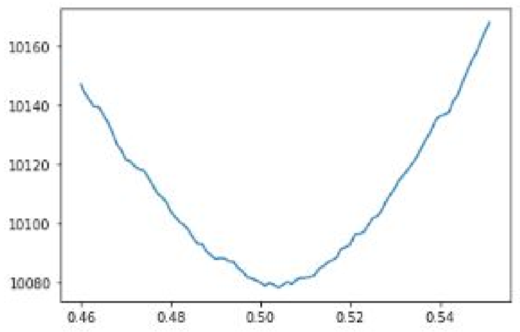

Figure 6. Value of the loss depending on the threshold on 200 images

Model	Loss	F1	Accuracy	Epochs	Number of images trained on
Best	BCE	0.874	0.935	200	9000
test 2	BCE	0.865	0.934	100	4000
test 3	BCE and Dice	0.852	0.913	100	4000
test 4	BCE and Jaccard	0.856	0.917	100	4000
Baseline	Cross-entropy	0.594	0.718	100	100

Table I
AICROWD RESULTS TABLE

baseline and that we trained it using only the first 100 base images. Even though we are satisfied with our results, there is always room for improvement. After a few tests with and without normalisation of the data and test sets, we assumed that it did not affect results considerably. Because our test were made on small data sets to avoid a long training time, it still could be something worth looking into. Because of the time needed to train the final model (around 8 hours on google colab GPU's) we decided to do the final big training without.
We also observed that the more training images we used, the better our score was. However, training time and the allowed RAM on google colab were a bottleneck on this so we could not afford more tests with much larger data-sets.

Moreover a deeper U-Net with other layers could have given us a better score. We also could have worked on alternative RGB representations. For example we could have trained our model on the HSV or HSL color scheme with the goal to give it more features to work and train on.
Finally, it's an idea at the boundary of the scope of this project, but for the road segmentation task, SAR (Synthetic-aperture radar) input could be a great help. SAR which is used to create two-dimensional images or three-dimensional reconstructions of landscapes can give us polarized input, which distinguishes clearly the road and the background by the input only.

VI. Other approaches

One of our first approach was using the pytorch based deep learning library called Fast.ai. It's simplicity was astounding for surprisingly good results. We tested many implementations to obtain the best possible model, however our tensorflow U-Net always gave a slightly better score. It however helped deciding which patch size we should train our model with. As it was easily changed, we could test it quickly. The first test was to copy the baseline approach and train the model on 16 by 16 pixels. The training was fast but results were worse than the ones of the given baseline model. By gently increasing the patches size we discovered a small improvement of the F1-score, and reached the paroxysm when feeding the model complete images directly. The training was a bit slower but it more than doubled our score. Fast.ai was a great way to experiment different data-sets, image pre and post-processing parameters, but in the end, it is our main tensorflow unet that took us above the 80 percent of F1-score.

VII. Conclusion

Working on this project was for us a great opportunity to develop our machine learning skills and helped us understand more precisely how neural networks works. Because we used different techniques to approach the problem, we now understand how to work with complex networks from a high and a low level perspective. Indeed, using pre-made libraries such as Fast.ai helped us understand the main lines of constructions of such models, and their training. And our final model implemented in tensorflow allowed us to understand the building of the model, layer by layer.

Although we believe that we could have gained a little edge by training our model for more epochs and on a bigger dataset, we can argue whether it was worth the environmental cost of the power needed. Indeed, training those models for 8 hours on google colab's GPU's not a cheap investment of energy and if in the firsts 10 to 20 epochs we achieved 0.8 of F1-Score, the 80-90 following epochs helped us 'only' reach 0.874.

References

[1] Nameirakpam Dhanachandra, Khumanthem Manglem, and Yambem Jina Chanu. Image segmentation using k -means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54:764 – 771, 2015. Eleventh International Conference on Communication Networks, ICCN 2015, August 21-23, 2015, Bangalore, India Eleventh International Conference on Data Mining and Warehousing, ICDMW 2015, August 21-23, 2015, Bangalore, India Eleventh International Conference on Image and Signal Processing, ICISP 2015, August 21-23, 2015, Bangalore, India.

[2] V. Sivakumar and V. Murugesh. A brief study of image segmentation using thresholding technique on a noisy image. In *International Conference on Information Communication and Embedded Systems (ICICES2014)*. IEEE, February 2014.

[3] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey, 2020.

[4] Knowledge transfer. U-net image segmentation in keras, 2019.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[6] Jerin Paul. Segmentation of roads in aerial images., 2019.

[7] Shuchen Du. Understanding dice loss for crisp boundary detection, 2020.

[8] Gabriel de Marmiesse. Jaccard loss on github, 2018.

[9] Guillaume Chevalier. Make smooth predictions by blending image patches, such as for image segmentation, 2017.