Machine Learning - Project 2
Road Segmentation

Raphaël Ausilio, Valentin Bigot, Valentin Karam *submission n° 110110*

***Abstract*—Since the advent of machine learning, segmentation and processing of images, more widely known as computer vision, has very quickly become a widely popular machine learning topic. Fundamentally, the objective of image segmentation algorithms is to gather a large number of images, and to assign a label to each pixel, in order to be able to automatically detect the features of an unknown image.**

I. Introduction

The aim of this paper is to build an algorithm able to decipher which pixels correspond to road or background from aerial satellite images. A set of 100 images was provided with their ground truth, each pixel being labeled as either *'Road'* or *'Background'*. We designed a pre-processing pipeline in order to increase the number of training images as the available number was low.
We then decided to train a U-Net model to perform the image segmentation. This type of neural networks has been performing especially well in low data availability conditions, mainly for biomedical imaging research [1]. For the EPFL-Machine Learning Challenge on AIcrowd, we also designed a post-processing algorithm, in which morphological operators have been encoded in order to eliminate miss-classified pixels as much as possible, as well as smoothing the resulting segmentation maps. For the computational part, Google Colab has been a great help to handle heavy calculations, where a large image set and many iterations were used.

Section II will show the stages of the data exploration and give details of the techniques used in the project. Section III will provide the results of our methods and their analysis. Section IV will provide a conclusion and analysis of the results obtained.

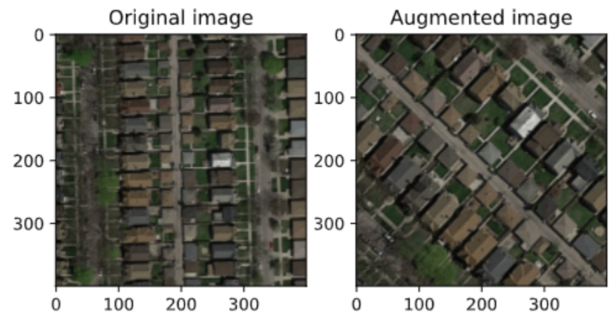
II. METHODOLOGY

A. Image set exploration

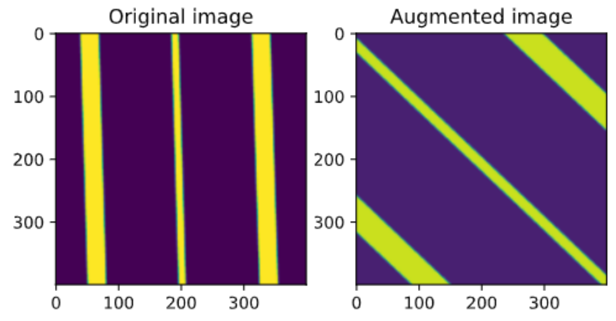
The set is composed of 100 satellite *images*, associated with their *groundtruth*. Both *groundtruth* and *images* are 400x400 pixels, and *images* are RGB.

The first objective is to explore the image set and to get familiar with it. The goal is to look for any flaw that could lead to problems during the training phase. Here are a few observations concerning this observation phase.

- There is really not enough data to feed a model, – 100 images is not enough – and hence it will not be very robust.
- Many vertical and horizontal lines are present, but almost no diagonal roads are present in the set – no more than 1% of the given images – . This means this type of road is not well represented in our dataset. As a consequence, diagonal roads will be very poorly detected.



(a) Raw image with its 45° augmented rotated image.



(b) Raw groundtruth image with its 45° augmented rotated groundtruth

Figure 1: An image and its corresponding groundtruth rotated by 45° during the image augmentation process.

B. Pre-processing and Data Augmentation

In order to have a large enough dataset, we first augmented the training set. We followed classic paths [2] and ended up with 9 data augmentation operations : Images were rotated by 5 different angles, – 45°, -45°, 90°, 180° and 270° – as well as flipped over 4 axes, namely vertical, horizontal and the two diagonals of each squared image.

Not only do we increase the size of the dataset, but we also create a greater number of different road angles, which are now better represented over the training set. Figure 1 shows an example of a raw image containing only vertical roads giving 45° diagonal roads after data augmentation.

In addition, we choose to add some noise to the raw image. Indeed, Gaussian, speckle and Poisson noises are believed to increase the performance and robustness of the model. [3]. Among those three, we choose the Gaussian noise because it is widely recommended considering that it mitigates the overfitting.

We hence have a total of **10 different transformations**, which means the dataset is multiplied by a factor of 10.

III. The U-Net Model, a highly performing segmentation neural network

The road segmentation task was performed with the use of a u-net model, which has been outperforming other deep neural network architecture in medical images segmentation [1]. It learns segmentation in an end-to-end setting which refers to the input of a raw image leading to a segmentation map and has shown very interesting results even with low training data availability.

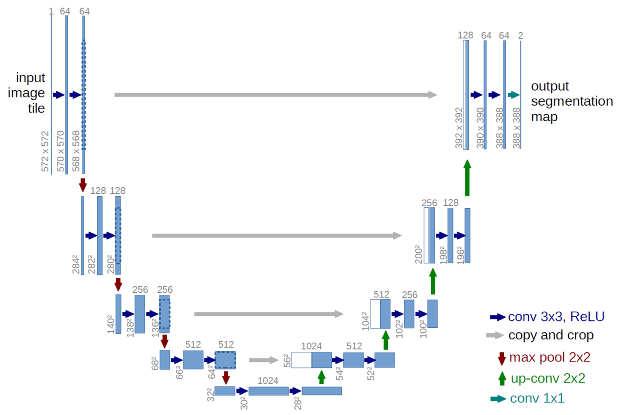


Figure 2: U-net architecture: an image is fed at the left part of the network, is processed following the operations described by the graph and the segmentation map is the right-side output. [1]

We provided the architecture graph of the U-net deep neural network where each blueish rectangle corresponds to a multi-channel feature map. Each blue arrow refers to a convolution followed by a non-linear activation function. Vertical red downward-facing arrows correspond to a max pooling operation. Vertical green up-ward arrows are de-convolution operators followed by a non-linear activation function and finally the green right-facing arrow at the right side of 2 corresponds to a 1x1 convolution outputting the segmentation map of the image.

A. The model

1) Contraction: The very power of this type of deep neural network architecture comes from the dimensionality reduction performed by the left side of the network which increases the density of contained information for each multi-channel feature map. It lowers the dimentionality of feature maps while keeping as much information as possible for each feature. This process is called spatial ”Contraction”.

- **Convolution Layers** The 2D convolution layers consisted in 32*n filters of 3x3 pixels. The n value was increased by a factor 2 after each pooling operator to increase the density of information contained at each feature point, respectively 1,2,4,8 and finally 16. Filters were initialized from a truncated normal distribution centered on 0 with a standard deviation depending on the number of units in the weight tensor. The same padding size between input/output maps was used. A batch normalization layer was then implemented before the use of LeakyReLu activation function with an alpha value of 0.25. This function is defined as

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

- **Max pooling** To down sample the multi-channel feature map, a 2D max pooling layer was implemented and consisted of 2 by 2 mask where only the maximal value of the square was kept.

A the end of this step, only a single feature channel vector remains and is ready to be processed by the right side of the architecture, which is referred as the ”Expansion” procedure.

2) Expansion: The expansion side of the architecture allows to recover the size of the input image to result in the segmentation map.

- **Convolution Layers** Again, 2d convolution layers were used and consisted in 32*m filters of 3x3 pixels. The m value was decreased by a factor 2 after each deconvolution operator, respectively 8,4,2 and finally 1. Filters were initialized from a truncated normal distribution centered on 0 with a standard deviation depending on the number of units in the weight tensor. The same padding size between input/output maps was used. A batch normalization layer was again implemented before the use of LeakyReLu activation function with an alpha value of 0.25.
- **De-convolution** To up-sample the multi-channel feature map, a 2D de-convolution layer was implemented and consisted of 2 by 2 mask with also the same padding size and filter initialization protocol.
- **Concatenation** Contraction and Expansion layers were concatenated to form the grey arrow depicted on Fig.2. Indeed, we did not proceed by cropping images which would have been necessary if the padding size of the convolutional layers led to smaller images than the inputs.

Finally, the segmentation map was obtained from the last convolutional layer by applying a 1 pixel 2D convolution constrained by a sigmoid activation function.

B. Training

Our model was trained with the use of Google Colab due to restrained computational local power. We were provided with a 12GB NVIDIA Tesla K80 GPU for the model training.

C. Optimization

The optimization method for this project was based on the Adam algorithm. Adam optimization is equivalent to a stochastic gradient descent that is regulated with an adaptive estimation of first-order and second-order moments. We justify

the use of such algorithm due to its high performance for Visual application of machine learning as well as fast overall computation. The loss function was binary cross-entropy which is widely used for pixel binary classification neural networks.
As the results are evaluated based on the F1 score, we chose to monitor the 2 metrics needed to compute it, namely Precision and Recall.

IV. Test set details

The model was hence trained on 400x400 pixels size images. The testing set available contained 608x608 pixels images so that an accurate image reconstruction was needed in order to correctly provide submission images. To do so, 4 squares 400 pixels-wide were processed in the neural network and accurately reconstructed at the end of the process. In the end, the image submitted was the combination of those 4 squares so that the first square prediction only remains in the 208x208 top-left pixels and the final square of 400x400 pixels was the bottom-right one. Bottom-left and top-right pixels were therefore 208x400 pixels wide rectangles of intermediary squares remaining from this reconstruction process.

V. Post-processing

Once the first results were collected, we realized that some errors in our predictions were recurrent and hence predictable. Moreover, most of the time, the width of the road is not respected.

It is therefore very likely that some morphological operators can erase this kind of error. But since the roads are never perfectly horizontal, vertical or oriented at 45°, most of the kernels will be useless on our actual data because they are chosen in advance. In order to design kernels that fit perfectly the angles of the roads, we used the **Hough Transform** feature extraction technique.

A. Hough Transform

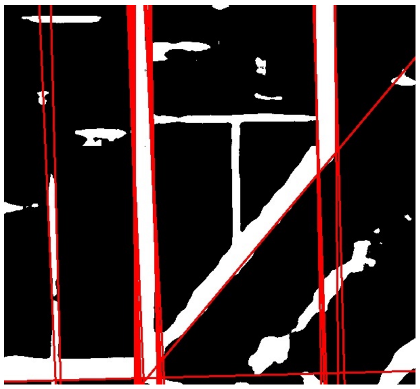


Figure 3: Plot of the Hough transformation algorithm on a binary prediction image. It matches lines on the edges of the road, and returns the tilt angles of the roads. In this case, the angles returned are 178°, 89° and 39°.

We used the Hough Transorm to extract the tilt angles of the roads from our predicted data :

First, the edges of the lines are isolated using a Kanny edge detection filter. Then the 'most matching' lines are stored, and their tilt angles are averaged to get the mean angle for each road.

On Figure 3, the detected lines are plotted on the predicted binary image. Three road angles have been detected (178°, 89° and 39°), so every parallel roads having these tilt angles are going to be processed.

If a road is not detected, it will simply not be enhanced i.e post-processing operations cannot harm and existing prediction image.

B. Filter kernels tuning

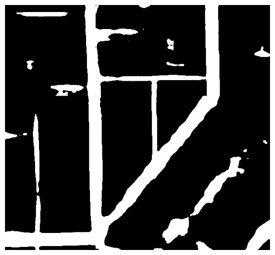
Once the angle of each road is extracted, the kernel can be tuned. Every tilt angle is assigned to a custom kernel, that has been design to warp . As a kernel is very specific to a unique tilt angle, a kernel is only applied on the lines its meant to wrap, and it ignores all others lines.

C. Morphological operations

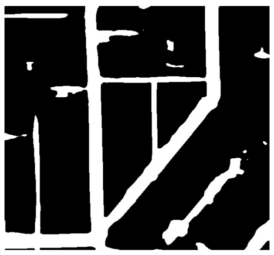
Once all the kernels are chosen, the morphological operations can be realized. The kernel are stretched out along the direction that they must follow.

1) An opening operation with the custom kernels, rejects roads that don't have the same kernels.
2) A closing operation with the custom kernels, fills the missing gaps between the roads.
3) The addition of the raw prediction image, because as our precision is very high, it is unlikely that a pixel labeled as 'road' is in fact a 'background'.
4) Finally, closing the resulting images with a small 10x10 round kernel to close gaps between roads that may be still disconnected.

On Figure 4 is shown the effect of the post-processing operations. We can see the gaps are better filled, and the roads are much more straight.



(a) Model predictions before post-processing



(b) Model prediction after post-processing

Figure 4: Effect of the post-processing operations on one prediction image

VI. Results

Our first model yielded in a high precision but a low recall on the training set, which meant that further improvement were possible. Our model was well design to find road pixels accurately but was missing some in the images. The following table display the results of the used models:

F1 score	Precision	Recall
0.3841	0.9899	0.2383
0.859	0.927	-
0.875	0.924	-
0.854	0.935	-

The first row of the table depicts results of a local prediction on the training set with only 50 epochs and 30 images. As we can see, the model yielded in a high accuracy but wasn't able to find all road pixel. This is due to a low training set size and few number of iterations. The dropout parameter was at 0.5 value. The second row of the table depicts a prediction on the test set where the model was trained on 100 images, for 100 epochs and a learning rate of 0.001. What can be discussed is that our precision was rather high for this trial, but again out recall value was not sufficient to yield to a good F1 score. For the final submission of this project, we decided to change the learning rate of our model, by dividing it by a 2 factor yielding to the last two rows of the table. The third row is our final model without the post-processing pipeline, followed by the same prediction maps post-processed. As we can see, our post-processing pipeline was not leading to sufficiently good results to be retained for the final submission due to a too high number of missed road pixels.

VII. Discussion

Our pipeline design was implemented to best tackle the challenge of pixel labelling of road on aerial satellite images. However, to improve the results, implementing further tools to augment the data size could have been possible. Indeed, mirroring techniques on edges, contrast and lightning functions could have been of great use to feed even more our model.
Furthermore, we could have better manage the training part of the model, by increasing the number of epoch while monitoring the F1 score on a validation set to find the best number of iterations for our model. Indeed, because the U-net architecture offers the possibility to label an image pixel by pixel, the recall metric is highly dependent on the ability of the model to learn features appropriately. Our final model was able to decipher road pixel properly from background, but was missing some diagonal road as well as low pixel wide roads between houses. Visual inspection of the predicted images and respective overlay shows that indeed, vertical and horizontal roads were greatly labelled but that our model was struggling with other angles of satellite images. It could also have been a great idea to balance our training set images to have a higher relative number of diagonal roads.
Moreover, even after data increases, diagonal roads remain underrepresented, and are therefore less well detected than straight roads. Unfortunately we did not have time left to improve this point properly, but it clearly remains an area for improvement to work on. Finally, while we implemented a post processing pipeline to take advantage of several tools to remove miss-classified results and smooth roads for the submission, we were not able to achieve a sufficient enough Recall score so that it was leading to slightly worse F1 score. However, we believe that a better training of the model could have been very well suited for the use of our post-processing pipeline.

VIII. Conclusion

The U-net model that we used was able to yield great results for the task it was designed to solve. This type of architecture offers great structure to perform image segmentation and has been widely used for such task. In combination with data augmentation, the rendering of this type of computer vision algorithm can be really improved, and classify much more complicated datasets that require much more accuracy.
In the future, an enhanced post-processing algorithm would have greatly improved our results, for example with more complete morphological operators.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," pp. 234–241, 2015.
[2] J. Brownlee, "Best practices for preparing and augmenting image data for cnns," 2019, https://machinelearningmastery.com/best-practices-for-preparing-and-augmenting-image-data-for-convolutional-neural-networks/.
[3] M. E. Akbiyik, "Data augmentation in training {cnn}s: Injecting noise to images," 2020. [Online]. Available: https://openreview.net/forum?id=SkeKtyHYPS