Machine Learning Project 2 : Road Segmentation

Simon Wicky, Gaultier Lonfat, Joachim Dunant

***Abstract*—The second project of CS-433 propose an image segmentation task, where the goal is to label roads from aerial images. In this report, we describe how we use different architecture to tackle this problem.**

I. Introduction

The goal of this project is to differentiate road and background, given an aerial pictures. The dataset consists of a hundred pair of picture and groundtruth, 400 by 400 pixels. Such a pair is displayed in figure 1. To tackle this problem, we implemented several neural network architecture and we then tested their accuracy.

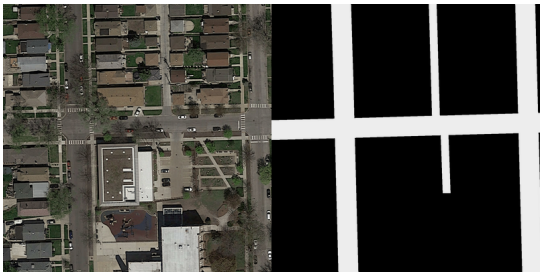

Fig. 1. Training image and its ground truth.

II. Method

In this part we will talk about our methodology to compute this machine learning problem. The first part is to understand the dataset and know what we are working with, if there are any suitable improvements to be made so that the neural network (NN) can be trained faster and with better results. Then, a major point is to choose a fitting NN to train our model. Multiple options are available and we will see which one we finally chose and why.

The evaluation method used for this project is the F1 score, which is given by:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

where $precision = \frac{TP}{TP+FP}$ and $recall = \frac{TP}{TP+FN}$ with TP being True Positives, FP False Positives and FN the False Negatives.

Note that changes between true and false positives will not change the score significantly, while lessening false negatives to have more true negatives will increase the F1 score notably.

III. Architecture

Convolutionnal Neural Networks (CNN) are known to yield significant results for image segmentation, since they take image spatiality into account. We first implemented a basic CNN composed of a couple of forward convolution layer and ReLU non-linearity, as well a CNN with an autoencoder structure. The results were irrelevant and are thus omitted.

A. UNet

Based on [1], we implemented a Unet, with a structure similar to 2. The size of each layer is adapted to fit our dataset. This network was trained with a Binary Cross-Entropy Loss, a learning rate of 10^{-4} and an Adam optimizer with a weight decay of 10^{-8}. The results are presented in section VI

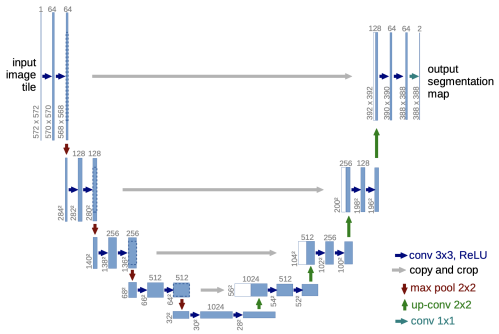

Fig. 2. UNet architecture

B. CNN with patches

The UNet architecture segments an image at a pixel level, but since the road are often quite large and straight, we can leverage this structure and segment at a coarser level. The image is split into a list of patches, 16x16 pixels, and fed to the CNN. Its architecture is displayed in figure 3.

This network was trained with a Binary Cross-Entropy Loss, a learning rate of 10^{-4} and an Adam optimizer with a weight decay of 10^{-8}. The results are presented in section VI.

```
ConvNet(
  (conv1): Conv2d(3, 32, kernel_size=(2, 2), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (relu): ReLU()

  (conv2): Conv2d(32, 64, kernel_size=(4, 4), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (relu): ReLU()

  (conv3): Conv2d(64, 128, kernel_size=(2, 2), stride=(1, 1))
  (relu): ReLU()

  (fc1): Linear(in_features=128, out_features=512, bias=True)
  (relu): ReLU()

  (fc2): Linear(in_features=512, out_features=1, bias=True)

)
```

Fig. 3. CNN architecture

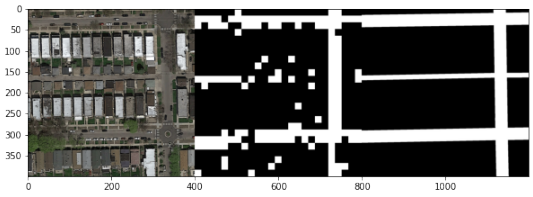

Fig. 4. Example prediction from CNN

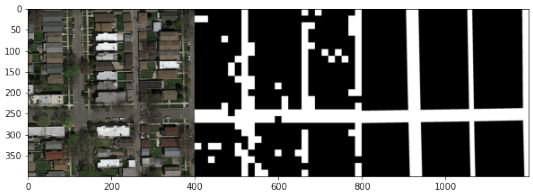

Fig. 5. Example prediction from CNN

IV. Preprocessing

A. Data augmentation

Since the provided data are quite few, we augmented the dataset by randomly flipping vertically and horizontally the given image-groundtruth pair. It is easy to see that such a task should be invariant to these transforms, thus producing new valid samples.

B. Image processing

To improve the results of the predictions, a possible way was to try doing some preprocessing on the images themselves as well before training the Neural Net. Many attempts were done, but two main ideas stood out as the most reasonable ones, even if the results after training were under our expectations.

1) Gaussian Blur: The gaussian blur sounded like a potentially good method of preprocessing, indeed it smoothes colors more uniformly all over the picture, making potentially bothering little details disappear, unfortunately, the CNN we use is very sensitive to these details and loses in accuracy after blurring (F1 score of 0.573)

Fig. 6. Prediction with gaussian blur, can be compared with figure 5

2) Pixels filtering: The other main idea, was to actually filter pixels that are completely unneeded, like grass, dirt. As the roads are mainly variants of gray, we kept all pixels where the difference between the three RGB channel is lower than a threshold, and set the rest to black, then we smooth the result to remove as many artefacts as possible. Like before the CNN is actually extremely sensitive and loses itself (F1-score of 0.419)

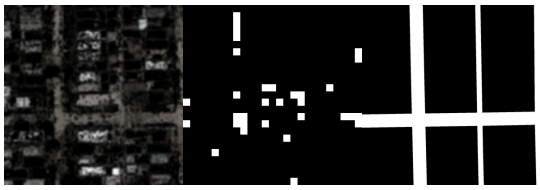

Fig. 7. Prediction with pixel filtering, can be compared with figure 5

V. Postprocessing

The postprocessing of the predictions is done on the CNN with patches and is used to remove noise and predict lines (horizontal or vertical), which is done using *morphology* from skimage's library. Firstly, we have to transform our patched image into one where each patch is a single pixel, so that *morphology* can be more efficient. Then, using a rectangular binary closure filter in the x and one in the y axis, we obtain two images that are noiseless and with respectively horizontal and vertical lines computed.

```
closed = morphology.binary_closing(start, selem=morphology.square(2))

opened_vert = morphology.opening(closed, selem=morphology.rectangle(5,1))
opened_hor = morphology.opening(closed, selem=morphology.rectangle(1,5))

reclosed_vert = morphology.closing(opened_vert, selem=morphology.rectangle(5,1))
reclosed_hor = morphology.closing(opened_hor, selem=morphology.rectangle(1,5))
```

Fig. 8. Horizontal and vertical binary closure

We can then simply use a binary-OR operation between those two images composed of 0s and 1s, before bringing it back to its original shape.

VI. Results

A. UNet

The UNet was trained for 100 epochs. The training loss and validation loss is displayed in figure 11. As expected, the size of this network combined with the small amount of

Fig. 9. Good prediction with postprocessing, can be compared to 5

Fig. 10. Bad prediction with postprocessing

UNet	CNN	CNN postproc	CNN gaussian postproc	CNN pix filter postproc
0.338	0.613	0.651	0.573	0.419

TABLE I
COMPARISON OF THE DIFFERENT F1 SCORES

data leads to overfitting, even with regularization. On the test set, this UNet obtained a F1 score of 0.339, which is quite bad. Some results were quite good, but other were less good. Figures 12 and 13 shows results obtained with the UNet.

Due to the poor initial result, we didn't apply any special preprocessing or postprocessing to this Neural Network.

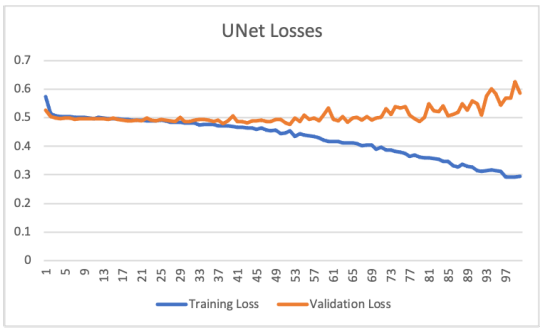

Fig. 11. UNet losses

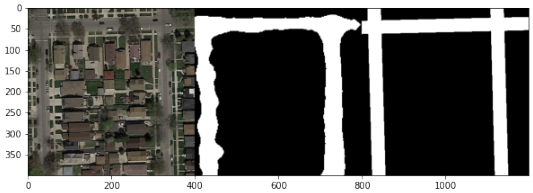

Fig. 12. Good prediction from UNet

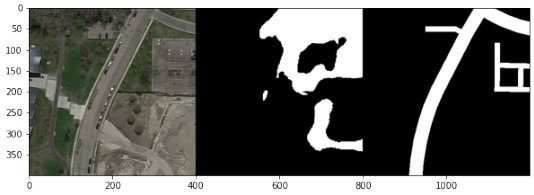

Fig. 13. Bad prediction from UNet

B. CNN

The CNN was trained for 200 epochs. The training loss and validation loss is displayed in figure 14. The training was quite slow, and didn't show significant improvement in the second part. We used only 100 epochs to obtain the following results, and to test the effect of the pre and postprocessing.

1) No processing: Without any form of processing, the CNN obtained a F1 score of 0.613. Figures 5 and 4 shows examples of obtained prediction. The nature of the CNN (the patches) cause the prediction to be "pixellated". To avoid this effect, we can leverage the structure of a road to postprocess these coarse prediction into a much better result.

2) Postprocessing: By doing the postprocessing over the predictions given by the CNN, the F1 score of our predictions rises slightly to 0.651, while the accuracy grows to 0.810. We can see that even though the precision rises greatly while the "prediction" of linear roads is working as intended, the F1 score is still not so good. This is explained because of the poor the prediction done by the CNN before this postprocessing.

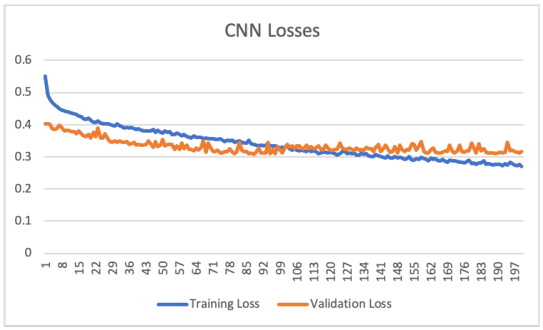

Fig. 14. CNN losses

VII. CONCLUSION

We tried two completely different approach for this problem. Segmenting on a pixel level with a UNet looked promising, but the lack of data had a big impact on the results. Another downside of this method is the lack of awareness of what we are looking for. Roads have a very recognizable pattern, and this particularity should be taken into account. Segmenting on a patch level lead to very interesting results. The postprocessing seems to greatly enhance the quality of the prediction, even if the score does not reflect this. We also

thought the preprocessing would have a huge impact, which it did not.

In future works, we could try to randomly crop, scale and rotate the dataset to augment even more its size. We could also try to train a neural network to detect occlusions by trees for example.

REFERENCES

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.