Machine Learning Project 2: Road Segmentation

Luca Barras (257916)

***Abstract*—In this project, the goal is to recognize the roads on satellites images coming from Google Map. To achieve this task, I use a convolutional neural network(CNN) that takes as input a satellite image and output the mask of the input image that contains the roads(white for roads and black otherwise). In this report, I use an U-Net architecture and try to optimize it to get the best f1-score as possible. To optimize the accuracy of the network, I use different techniques like data augmentation, regularization and try different numbers of layers in my CNN.**

I. Introduction

In this project, I try to classify patches to say if they are a road or not a road. I don't have directly the patches, so I must to split the trainings images(the satellites images with their groundtruth) in patches of size 16x16 pixels. If the patch contain more than 25 percent of road pixels so I classify it as road(white or 1) otherwise as not a road(black or 0). The training set contain 100 pairs of images(original satellites images with their groundtruth) of size 400x400 pixels. The testing set contain 50 images of 608x608 pixels, so we must classify 72200 patches. In the section II and III, I will present the baseline model and after the implementation of my fully convolutional network. In the section IV and V, I will speak about some technique that improve the accuracy of my network like data augmentation and the importance of the size of the input of the model. And I will talk about some other networks architectures I tried but that was less promising than the network architecture I used. And finally in the parts VI and VII, I will talk about my implementation and the hardware used to train the model and show the results of the different models.

Figure 1. Example of an image of the training set with its corresponding groundtruth

II. Baseline Model

The baseline model is a simple convolutional neural network with respectively 32 and 64 5x5 filters and 2 fully connected layers with regularization for the binary classification. It uses ReLU as activation functions and a fully connected layer of 512 neurons. This model predicts directly each patch of 16x16 pixels. This method yield to a f1 score of 0.804 on AICrowd. This method has a serious limitation, the patches are classified without taking into consideration its neighborhood. Indeed a patch of 16x16 pixels does not contain enough information to handle case like trees or cars on the road and cannot well capture the continuity of the roads. Another limitation is that the model is not enough complex to extract all the features needed to classify the different patches and this model probably suffer of underfitting. Finally, the training set is too small and must clearly be augmented.

III. Network architecture

I used a fully convolutional network with a U-Net architecture [1]. You can see its architecture on the figure 2.

So in the first part, the encoder part, I use a series of convolution with a kernel size of 4, a stride of 2 and a padding of 1(allow to divide the image by exactly 2 at each step). Each convolution is followed by a ReLU activation, a batch normalization and a Dropout of 0.2. In this part, at each layer, we increase the number of features maps.

In the second part of the network, the decoder part, I use a series of transposed convolutions with a kernel size of 4, a stride of 2 and a padding of 1. This configuration allow to increase the size of the features map by a factor of 2 exactly. And additionally, I concatenate the encoded input at the same levels follow by a convolution like shown in the figure 2. And finally, I ended with a convolution that map from 16 to 1 channel followed by a sigmoid function to get the probability of each pixel to be a road.

In this architecture I used ReLU activation function because it is an activation function that works very well in practise and after some experimentation, this is the activation that give me the best results. I used transposed convolutions instead of upsampling because it allows the network to learn the best upsampling methods. It has been tested in many papers. I also use dropout and batch normalization for regularization. After some experimentations, I choose to use a dropout of 0.2 after the convolutions. In addition to reduce overfitting, dropouts also allow to decrease the training time. And finally, batch normalization is used to normalize the inputs of each layer by adjusting and scaling the activations which results in a more stable training [2].

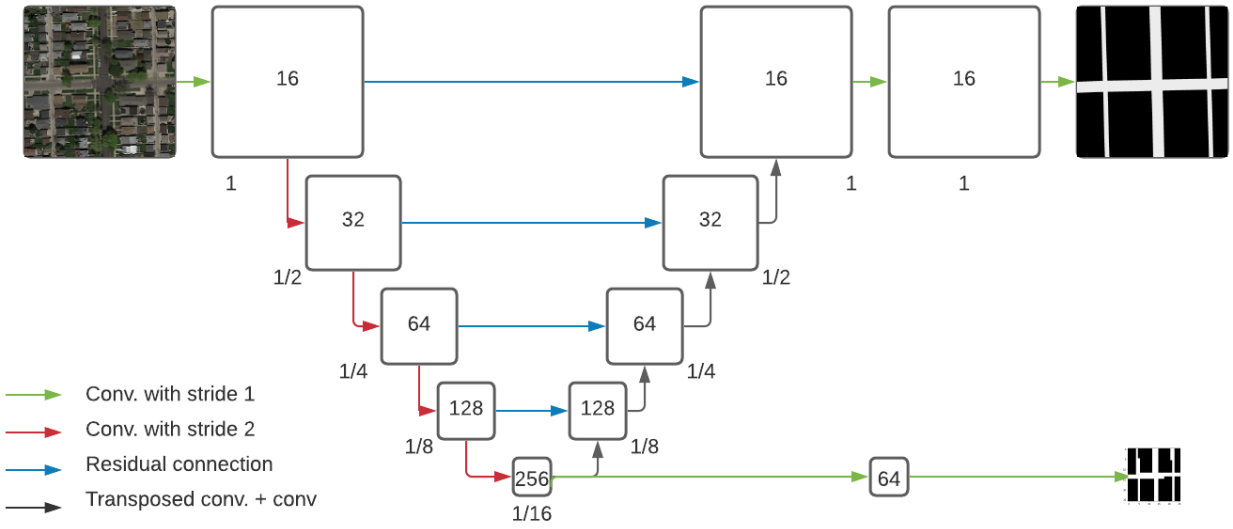

Figure 2. Architecture of the model. Inside the boxes this is the number of features maps and below the boxes this is the size ratio of the features maps with respect to the input. This model output two images, the first one is the mask of input images and the second one is the patches representation of the input images.

With regard to the loss function, I use the binary cross entropy, that is the more appropriate for the tasks like classification. I used also an objective loss that compare the 16x16 pixels patches with the second output of the network (the bottom output in the figure 2). This output is directly the corresponding groundtruth patches of the input images (image / 16 $->$ 16x16 pixels patches). I weight these two loss functions with 2 weights λ_1 and λ_2 to try to better generalize my network.

For the optimizer, I used Adam with a learning rate of 0.001. And finally, I used batches of 20 to reduce the training time.

The network take in input a image of size 304x304 and output the corresponding mask of size 304x304 too. The mask contain for each pixel, the probability that the pixel is a road. This network outpout also the classification of the 16x16 patches of the input image as explained above.

IV. Data Augmentation

Deep neural networks are very powerful but to be effective, they need a huge amount of data. So in general, more and more they have data, more and more they are accurate. In your case, we have only 100 images which is far from enough, and we have probably great chances of overfit to them. So we need to increase the number of images.

A. Flip and rotations

Convolutionnal neural network are translation-invariant but they are not invariant to rotations and flip. By looking at the images in the training set, we remark that there are a lot of horizontal and vertical roads but few of diagonal roads. So applying a rotation of 45 degree seems to be a good idea. So I increase the number of images by applying some rotations and flipping (horizontal and vertical) on the fly to each sample and this allowed to greatly improve the accuracy of the model.

I could have use some other data augmentation methods like brightness or shearing but if we look the test images, we have the same color shades as the train images. So I decide to not use these methods because the resulting samples would be too different from the ones in the testing set.

B. Other datasets

I tried to find other datasets containing images of road map with their corresponding mask but all the datasets ([3] for example) I found were too different from the images in the test set. So add them would only increase the training time and not really increase the accuracy of the results. So I decided to not use them.

I have also modified some groundtruth images because they were not enough accurate and sometimes even false. So if the dataset was bigger it would not have been a problem but in our case we have a small dataset, so learning with some wrong groundtruth can pose a problem.

V. Context of image

16x16 pixels patch are not sufficient to classify roads. Indeed, to classify a road, we need to capture its context to

better classify it. In capturing the context, the network can learn some proprieties of roads like continuity and structures. For example if we take a road that passes under a bridge, use only a 16x16 pixels patch will not very well classify while with a bigger context, the network will understand that the road is continue and so will probably pass under the bridge. So to solve this problem, I used a network with Unet architecture that take in input a full image and not just a patch of 16x16 pixels. After some experimentation, in a first time I remark I have better results than the baseline model but not as much I expected. The problem is the size of the images of the training set are not the same size than the images of the test set and so the network has some difficulties to predict the masks. By analysing the images of the training set and test set I remarked that the scale in the images was similar but the images of the test set use just a wider area. To counter this problem, during the training, I crop randomly the images to have images of size 304x304. And to generate the groundtruth of the test set, I split the test set images in 2x2 patches of 304x304 pixels and put this 4 patches in the network. After this I reconstruct the full groundtruth by merging the 4 batches.

Figure 3. The 2x2 patches of size 304x304 pixels of an image of the test set. Each one of these patches are sent to the model and then remerging to form the final result.

VI. Implementation

I implemented the model with the help of PyTorch. For the training, I used a learning rate of 0.001 for the 700 first epochs and then a learning rate of 0.0001 for the 250 last epochs with the Adam optimizer. To speed up the training time, I used batches of size 20. I used $\lambda_1 = 1$ and $\lambda_2 = 0.5$ for the weights of the loss functions.

Concerning the hardware, I used a NVIDIA GeForce RTX 2070 8GB to speed up the training on the batches. The time to train the model took about 3 hour.

VII. Results

The performance of the different models are shown in the Table 1. We can see that Unet and Data augmentation increase significantly the performance. In figure 4 we can see the results of my model on some test set images. We can see that the model perform very well and the results are very close to the ones we an expected from a human.

Model	**F1 score**
Baseline	0.804
Unet	0.842
Unet + Data augmentation	0.894

Table I
F1 score of the different models used during the project.

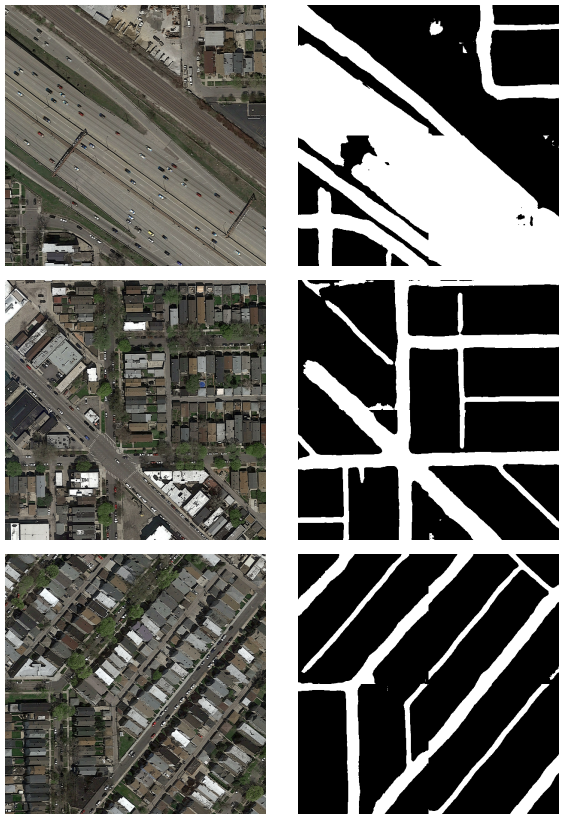

Figure 4. Some results of the test set. On the left the image of the test set and on the right the prediction pixel wise with the Unet model + Data augmentation.

VIII. Implementation

IX. Conclusion and Limitations

This project showed that the size of the training set play a role very important. Indeed, neural networks need a lot of data to perform well and not overfit. So data augmentation when we have a small training set is the best solution. This project also showed that training with images that have the same size than the test set images give much better results than training with a different size.

For future work, I thought about some solutions to improve even further the performance of the model. To find a dataset with the same kind of images is probably the better

option to increase the results. It allows the network to less overfit and see some case that was in the test set but not in the training set. Use transfer learning could also be a good option.

REFERENCES

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

[2] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[3] V. Mnih, "Machine learning for aerial image labeling," Ph.D. dissertation, CAN, 2013, aAINR96184.