Road Segmentation

An experimental approach to Machine Learning in computer vision

Ugo Damiano, Eric Jollès, Samad Durussel
Department of Computer Science, EPFL Lausanne, Switzerland

***Abstract*—In the many applications of Machine Learning, computer vision is one of the most exploited. These problems have given rise to several techniques such as Convolutional Neural Network (CNN) and their different variants. Here we propose to explore a known problem which is the one of segmentation and bring light on how the different techniques developed in the past years improved such models.**

I. Introduction

Image segmentation is an image processing operation that aims to label certain parts of an image according to predefined criteria. It is used in many domains such as biomedical imaging [1], and recognition tasks [2]. Faster processors and parallel computations on GPUs, allow new machine learning techniques to answer these problems in a reasonable time.

This paper is an overview of different road segmentation techniques and focuses on Convolutional Neural Networks which became the state of the art technique for image segmentation [3].

II. Data Exploration

The available dataset contains 100 aerial views images of 400x400 RGB pixels with their ground truths, which are black and white images showing if a pixel is a pixel or a road (white pixel) or background (black pixel) (see Fig 1).

We will classify blocks of 16×16 pixels , by giving label 1 if the proportion of background pixels in the block is greater than 0.25 and label 0 otherwise.

III. Metrics

Through this paper we will use two performance evaluation metrics:

- the **validation loss** : the value of the cost function of our cross-validation data. The loss function used here is the binary cross entropy. Which can be written as :
  $$-\frac{1}{N}\sum_{n=1}^{N}[y_n log(\hat{y}_n) + (1-y_n) log(1-\hat{y}_n)]$$
  Where $\hat{y} = \frac{1}{1+e^{-f(x)}}$ where f is the prediction function and y is the ground truth.
- the **f1-score** : the harmonic mean between precision and recall. Where :
  - precision : number of pixels correctly classified as roads divided by the total number of pixels classified as roads.
  - recall : number of pixels correctly classified as roads divided by the total number of pixels that should be classified as roads.

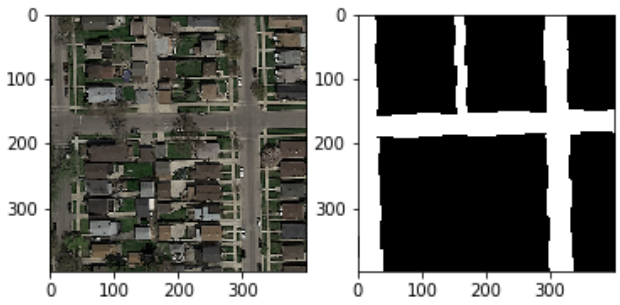

Fig. 1. Aerial image (left) with its corresponding ground truth image(right) given in the train data

IV. Models

A. Logistic Regression

We first tried a basic machine learning model : the logistic regression. It took as input a batch of 16×16 pixels. We obtained really poor results with this model as we can see in Fig. 2. It is due to the lack of context of each batch, which knows nothing about its neighbors. We thus decided to use more advanced machine learning models: Convolutional Neural Networks (CNN).

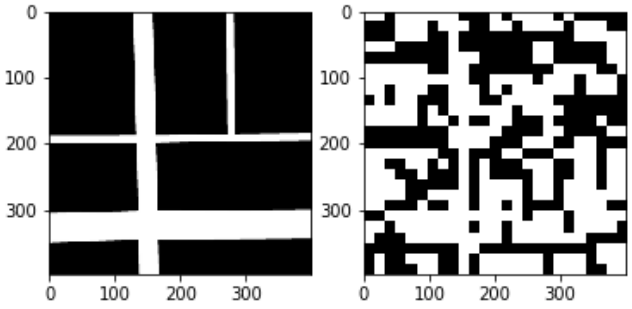

Fig. 2. Ground truth (left) and prediction (right) with the logistic regression model

B. Convolutional Neural Network (CNN)

We used as our reference model the one given with the project statement which is a variant of a CNN. CNNs have been proven to be very effective in areas of classification and segmentation. It contains:

- A convolution operation : it computes the output of neurons that are connected to local regions in the input.

- An activation function : it is the non linear transformation processed over the input signal after being passed in the convolution operation.
- Pooling : it reduces the dimensionality which enables us to reduce the number of parameters, which both shortens the training time and reduces overfitting. Extracts the most important elements of the input.
- Full connection : it wraps up the CNN architecture to create the prediction.

C. U-Net

The main issue with a classic CNN in image segmentation is the lack of context of neighboring pixels. We thus implemented a U-Net neural network [6]. The U-Net architecture is built upon the CNN. U-net is symmetric and has skip connections between the downsampling path and the upsampling path to provide local information to the global information while upsampling. The architecture is depicted in Fig. 3.

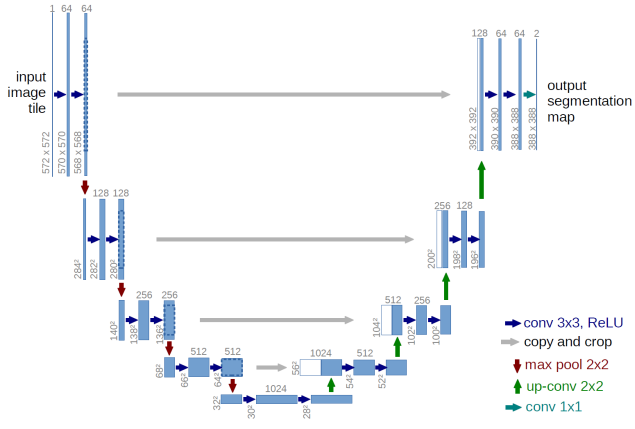

Fig. 3. U-Net architecture from [6]

V. Optimizations

Throughout this project, we use many optimizations that we will list in this section.

A. Learning Rate and Early Stop

In order to find the best learning rate for the training of our models, we tried to use grid search between 10^{-2} and 10^{-6}. We observed that the best trade-off between training speed and model performance is to use a learning rate of 10^{-4}. In order to achieve better results, we decided to use an adaptive learning rate, which starts at 10^{-4} and is divided by 4 every time the validation loss is not decreasing monotonously (see fig 7 later), in order to get better granularity for finding the best possible weights for our models. In addition, we use early stop. We stop the training of the model as soon as we are convinced that the loss over our validation set will not decrease anymore, since further training will only lead to overfitting.

B. Activation function

In Neural Network models, ReLU is the the standard activation function [4] since it facilitates training. It is defined as $f(x) = max(0, x)$ and thus simply truncates negative values to 0. The main issue is called "dying ReLU" problem and occurs when we always have neurons with 0 weight value, we then have a null gradient which blocks the learning. Leaky-ReLU tries to keep advantages of ReLU but overshoots the "dying ReLU" problem by having a small negative slope. It is defined as $f(x) = max(\alpha x, x)$ with small positive α.

C. Image augmentation

A downside is that such a network has a large number of parameters and so is extremely data-hungry. Our dataset is not large enough to train a good model. However we can generate new samples from old ones. We thus decided to make multiple image transformations with 45 degree rotations and zooms.

D. Regularization

As for standard regression and classification tasks, we can use regularization terms. This avoids exploding gradients and also overfitting.

E. Dropout

Dropout is a method both to avoid overfitting as well as to do model averaging [5]. In this technique, at every training step, we decide for each node i in layer l with probability $p_i^{(l)}$ whether to keep this node or not for the computation.

F. Predictions dimension

Some of the models we implement are trained on smaller images than the ones we need to predict. In order to adapt the prediction size to the training size, we perform the prediction on the four corners of the picture. We combine them by taking the maximum of the overlapping surfaces.

G. Post-processing

The classification task is not trivial, since we can have trees, cars, building shadows over roads, but also parkings which look like roads. We thus observed that in general our models often have incomplete or isolated roads. We thus decided to solve this issue with image processing techniques. We applied convolution filters in order to identify broken horizontal, vertical or diagonal lines and to repair them. We also identified isolated pixels in order to remove them from the prediction as can be seen in Fig. 4. With this technique we obtained some improvements, but not as we expected. Since our model gives noisy predictions, our algorithm sometimes tries to complete roads that do not exist, which reduce the algorithm utility.

VI. Results

In this section we detail how each trial improves over the previous one and which optimization are brought. All results are summarized in Fig. 5.

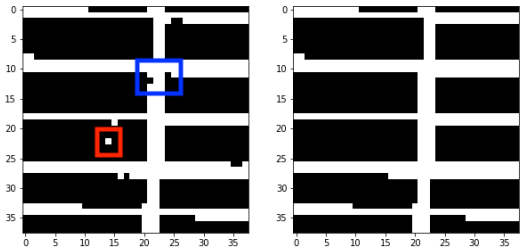

Fig. 4. Prediction (left) and corresponding post-processing result (right). Isolated road is highlighted in red; incomplete roads are highlighted in blue.

Model	Epochs	F1-score
CNN baseline	100	0.606
U-Net baseline	50	0.758
Leaky ReLU	50	0.741
Leaky ReLU	100	0.655
5x5 kernel	59 (early stop)	0.733
5x5 kernel and augmentation	39 (early stop)	0.835
6x6 kernel and augmentation	17 (early stop)	0.793

Fig. 5. F1-scores on the test set for the different models (on AIcrowd). "Early stop" means that the model stops learning when it reaches a loss plateau.

A. U-Net baseline

We implemented a state of the art U-Net as explained in Ronneberger et al's paper. The down-sampling path is made of 4 layers which are composed of 2 layers with filters of size 3x3, activation function ReLU and after both a max Pooling. A final layer similar to the previous ones but without max Pooling is applied. A dropout is performed on the last two layers. The up-sampling path is just the reverse of the above mentioned without dropout. We achieve an F1-score of 0.758 on the test set.

B. Leaky ReLU

We tried to change the activation function of our model from ReLU to leaky ReLU with $\alpha = 0.01$. We get worse results for the same number of epochs. Why should we expect such a result ? The model tries to improve the classification of already correctly classified samples. Due to our lack of diversity in our training set, this leads to quicker overfitting of our model, thus giving worse results than plain ReLU.

C. Filter size

On a more abstract level, we could think that increasing the filter sizes might be a good idea. In fact, determining that a pixel is a road relies a lot on what are its neighbor pixels. We observed that 6x6 filters gave poor results with different learning rates. As we can see in Fig. 6 it already overfits after a few epochs and thus provides a weak model. The validation loss already starts to increase after 50 epochs and we get a f1 score 0.4849.

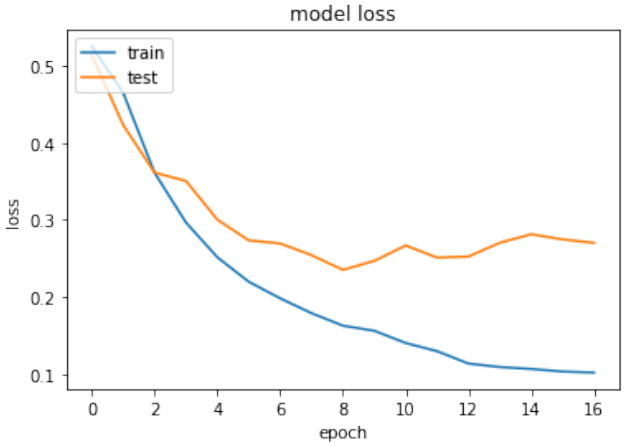

Fig. 6. Validation loss evolution over epochs for a kernel of size 6 with image augmentation

The best results we obtain are with a filter of size 5x5.

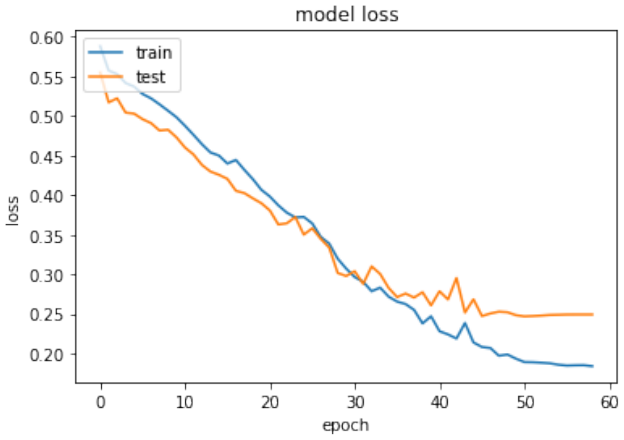

Fig. 7. Validation loss evolution over epochs for a kernel of size 5 and no image augmentation

D. Image augmentation

We need a lot of samples to train the huge amount of weights we get. Also, by a quick inspection we can observe that the training set lacks actually some specific samples. In fact the test set contains a lot of pictures with roads running diagonally, whereas our training set contains very few of those. By applying operations such as 45° rotations, we can generate those missing images, but we will have to pad the new images. To do so we perform a reflection on the border of the samples.

In each epoch we train on the whole augmented dataset which provides us around 10 times more samples. Hence an epoch with the augmented set trains the model more than one epoch with the non augmented set. As we observed in Fig. 8 the model overfit after 20 epochs and only after 60 in 7. Hence running the algorithm with image augmentation is similar to a model without augmentation for 200 epochs.

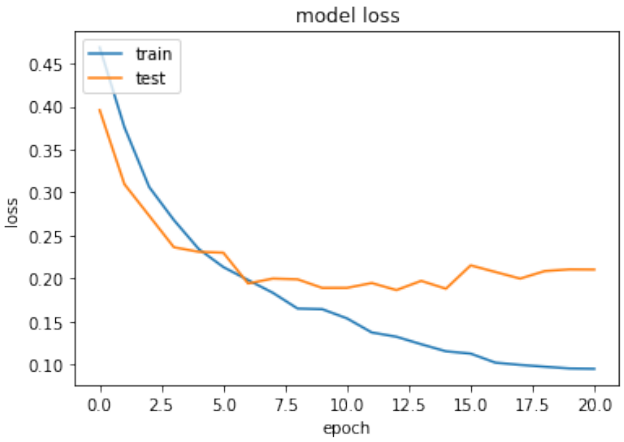

Fig. 8. Validation loss evolution over epochs for a kernel of size 5 and image augmentation

VII. Conclusion and future work

We have seen how the different optimizations we have tried improved the model. And they all make perfect sense. The model we proposed in section IV-A didn't use the fact that we were working on pictures. Indeed it didn't extract the maximal correlation by using filters and didn't try to rely on its neighbours, so we concluded that CNNs are a better approach to image segmentation.

The most prominent aspect of our research is that we have better training with image augmentation, since it provides a lot more material to the model for its training. We wanted to avoid overfitting as much as possible, but complex models such as U-nets need a lot of training data, and even though we think that increasing the kernel size is a good approach, we lack even more training samples.

Another approach would be to use road maps to improve the predictions. This relies more on the task than on the model itself but could still be explored. We could also increase the number of images created with augmentation, and vary even more the transformations. However this can cause two problems to arise in our case. The first is the memory requirements, that also increases with our data augmentations, if we try to store everything in RAM for faster training. The second problem is memory access, as we would need faster memory accesses in order to keep a realistic model training time, if we try to load data dynamically during training.

All in all our predictions could be greatly improved, though we still get some nice results with our current approach, as seen in Fig. 9.

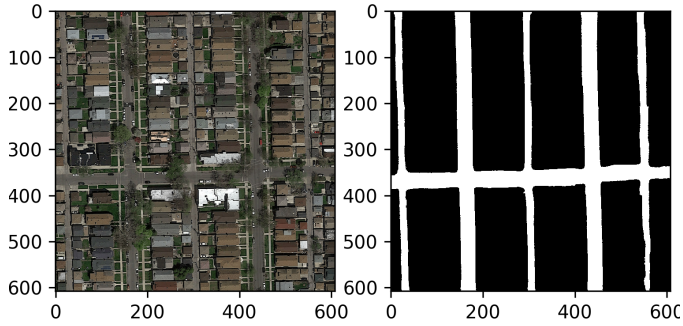

Fig. 9. The best road prediction with our best model

References

[1] *Medical Image Segmentation: A Brief Survey.* Ahmed Elnakib, Georgy Gime'farb, Jasjit Suri and Ayman El-Baz

[2] *Learning Aerial Image Segmentation from Online Maps.* Pascal Kaiser, Jan Dirk Wegner, Aurélien Lucchi, Martin Jaggi, Thomas Hofmann, and Konrad Schindler

[3] *Fully Convolutional Networks for Semantic Segmentation.* Jonathan Long, Evan Shelhamer and Trevor Darrell,

[4] *Deep Sparse Rectifier Neural Networks.* Xavier Glorot, Antoine Bordes and Yoshua Bengio (2011).

[5] *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov.

[6] *U-Net: Convolutional Networks for Biomedical Image Segmentation.* Olaf Ronneberger, Philipp Fischer and Thomas Brox.