Road segmentation

Estelle Chabanel, Charlotte Sertic and Servane Lunven
Machine Learning , EPFL Lausanne, Switzerland

***Abstract*—This paper provides different methods to perform road segmentation of satellite images, the purpose being to classify them into road and background segments. Deep learning provides efficient methods to tackle this problem. Three models are implemented and compared: a traditional convolutional neural network, a U-net and a mean of different models, the latter providing the best result, with a F1-score of 0.913 on the submission platform AIcrowd.**

I. Introduction

Road segmentation is the process by which a satellite image pixel is partitioned into two subgroups: road and background. Reducing the complexity of the image, it is used in different fields such as map drawing or GPS implementations. The different shapes of road and their similarity with rail roads, parking lots and rooftops make the task hard.

In this project, three different neural networks have been implemented to answer the question: a simple neural network, a U-net and finally a mean of different models. Their architecture and choice of parameters will be described in the paper, as well as the image preprocessing. These studies are performed on a training dataset composed of 100 images and their corresponding masks. The final scores of our models are obtained with a test dataset containing 50 images.

II. Data Preprocessing and Augmentation

The training set used for the model training is composed of 100 RGB satellite photographs and their corresponding black and white groundtruth images of size 400x400 pixels. A pair of images of this training set is presented in figure 1.

Fig. 1: A set of image-groundtruth

However, 100 pairs of images-groundtruths are not sufficient to train an efficient neural network model, therefore more data is needed. For each model, before training and transforming the images, these latter are augmented.

The data augmentation is done by applying image transformations on the original training set (both the satellite image and its corresponding groundtruth). These transformations are chosen such that the model is prepared for any kind of images. The purpose is to create the masks corresponding to the test images. By looking at them, one can notice that the test and training images share similar properties, except their size which is 608x608 pixels for a test image against 400x400 for a training one. One can also notice that the roads of the test dataset have every kind of orientations (vertical, horizontal, diagonal and curves) while the ones of the training images are more often vertical or horizontal. To solve this problem, different rotations are applied to the original dataset.

To have diagonal roads and roads of every kind of orientations, an additional set of data (image + groundtruth) is created by rotating the first one by an angle of 45°. Rotating the original set by four random angles in the interval [0°, 180°] creates four additional datasets and diversifies the road orientation. For each rotation, the border condition is a reflected one to keep images of the same size and eliminate any black border.

A sixth set is created by applying some horizontal and vertical flips to the images.

Then, one can also decide to add noise to the original dataset in order to prepare the model for bad quality satellite images. Here, this is done by applying a Gaussian Blur of kernel 5x5 to the original set. Finally, two last sets of data are created by modifying the brightness : brightness is both added and withdrawn from the pictures to simulate images taken under different weather and time conditions, or simply worse quality images.

The performed transformations on the satellite image in figure 1 are visible in the figure 2 .

In the end, the training dataset is composed of 1000 pairs of images-groundtruths.

The selected data are then divided in patches of different sizes, sizes that will be decided according to their efficiency. The images are split into patches, because these are more homogeneous than the complete image and will therefore be more accurately segmented in the neural network [12] . The size of these patches have to be determined to fit both the training and the testing set. This is why only three patches sizes will be tested : 16, 128 and 256. For the 128, the training images have to be cropped from 1 pixel, but the testing images stay intact.

For the training, the patches are finally split into two groups: a training and a validation set of respective size 80% and 20% to train the model in order to find the best parameters.

(a) Flips (b) Rotation of 45° (c) Random rotation 1

(d) Random rotation 2 (e) Random rotation 3 (f) Random rotation 4

(g) Gaussian Blur (h) Lightening effect (i) Darkening effect

Fig. 2: Augmented images for the original satellite image 1

III. Neural networks models

A. Traditional convolutional neural network

First and foremost, a traditional convolutional neural network is implemented to get familiar with the task of segmentation and define the next steps necessary to improve the model. For the remaining of this paper, this model will be denoted by its initials CNN.

In order not to loose the context surrounding a 16x16 patch, a window sliding method is used. The CNN will thus classify the 16x16 patch in the middle of the window, according to its context. The size of the window 80x80, has been chosen such that enough context surrounds the patch and that it is not computationally expensive. Since the window is larger than the patch, to be able to classify the patch on the edges, one needs to pad the images. For this the 'reflect' mode is used, so the outer edges are padded with their own reflection as visible on the image 6 in appendix.

To solve the problem of the memory consuming dataset, the model is trained by batch. Which means that for each epoch, a certain number of windows and their corresponding patches are randomly extracted to feed the neural network. The batch size which gave the best results is 256. The generator is run in parallel to the model. The architecture of the convolutionnal neural network is displayed in table I.

Layers	Parameters
Input	80x80x3
Conv2D + Leaky Relu	filters=64 and kernel=3x3
Max pooling	poolsize = 2x2
Dropout	$p_{drop} = 0.2$
Conv2D + Leaky Relu	filters=128 and kernel=3x3
Max pooling	poolsize = 2x2
Dropout	$p_{drop} = 0.2$
Conv2D + Leaky Relu	filters=256 and kernel=3x3
Max pooling	poolsize =2x2
Dropout	$p_{drop} = 0.2$
Fully connected	128 nodes
Leaky Relu	
Dropout	$p_{drop} = 0.5$

TABLE I: Architecture of the convolutional neural network

The softmax activation function is used in the output layer. The best result obtained with this model is 0.886. There is room for improvement.

B. U-Net

Next, to obtain better results, a fully convolutional network called U-Net has been implemented. Indeed, a U-Net is an efficient network used for image segmentation as it allows the classification of each pixel of the image.

The U-Net architecture is separated in two phases as visible on the scheme of the network in figure 3. The first one, called the encoder is a stack of convolutional and max pooling layers that capture the context of the image. The second one, the decoder, contains transposed convolutions which enable precise localization.

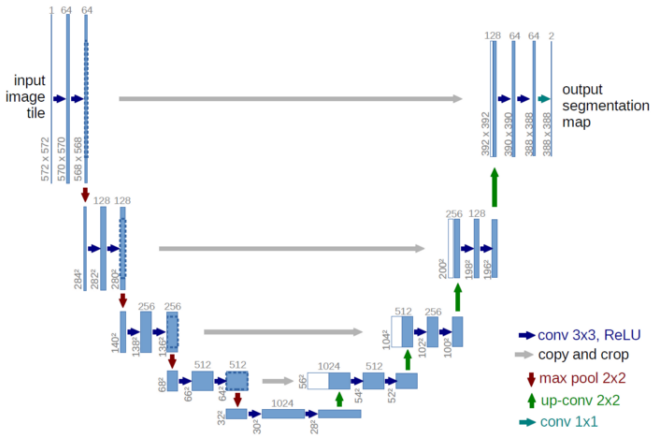


Fig. 3: Architecture of the U-Net network [1]

Thus, for an input image of the desired patch size, two convolutions are applied, followed by the activation function leaky ReLU and a 2x2 max pooling. After each convolution the images are normalised, to avoid overfitting. This process is repeated several times, each time, dividing the size of the image by two and multiplying its number of channels by two.

Once, the downsampling is finished, we enter the upsampling or decoder path : a transposed convolution is applied, followed by a concatenation with the feature map of the encoder at the same level to get precise location. There is again two other convolutions followed by the activation function leaky ReLU, with batch normalisation after each convolution. This process is again repeated several time, each time doubling the size of the image and dividing its number of channel by two to regain the original size of input. A final convolution with kernel 1x1 and the activation function *sigmoid* gives the final output : the class of each component.

C. Multi-resnet

The final model implemented is a combination of predictions of multiple models, called the ensemble method. This method is used to increase the performance of the models by reducing the variance [2]. As the models are trained by stochastic gradient descent with a batch size of 16, a high training variance is created. Through the combination of the different models, a bias is added which counters the variance of the training models [2]. Therefore, the models must be chosen carefully to be good but also to be different, in order to create different errors and thus be more robust[2].

Three different variations of the resnet model are used in this ensemble model. The resnet model was developed in 2015 by Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun and stands for residual neural network [3]. The residual neural network contains residual blocks that are connected with a skip connection, as shown in figure 4. This method is used to solve the problem of vanishing gradient that arises with more and more layers, by allowing the information of the gradient to flow through the skip connection[4].

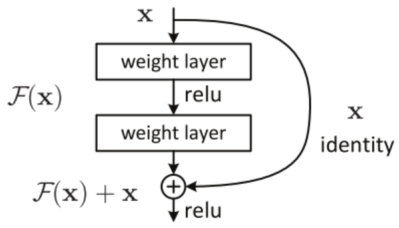


Fig. 4: Residual block [4]

The ensemble model combines Resnet 34 and Resnet 50 which are two variations of Resnet with respectively 34 and 50 layers. The models are initialised with the activation function *sigmoid* and with 1 class only, as this is a binary image segmentation. In addition to these two, a SeResnet 50 model is applied. It is a Resnet model with SE blocks, that stand for "squeeze and excitation". Re calibrating features through each SE block is used to boost the representational power of the model [5].

IV. Training and first results

The results for the different implemented models are presented below. For all of them, it is necessary to find the hyper parameters in order to obtain the most efficient model possible. These hyper parameters are: the learning rate, the patch size, the filter number and the number of epochs and will be presented for each model. The CNN model being less efficient than the others, no big study has been carried on the hyper parameters, that are already given above.

All models are trained with the *binary cross-entropy* loss function, shown in equation 1[13]. This loss function calculates the performance of a model whose output is between 0 and 1, which is the case since all models have the activation function *sigmoid*. The binary cross-entropy loss converges when the predicted value $p(y_i)$ is close to the real value y_i.

$$Loss = \frac{1}{N}\sum_{i=1}^{N} y_i \log(p(y_i)) + (1 - y_i) \log(1 - p(y_i)) \quad (1)$$

All models are optimised with the *Adam* optimiser, which is a stochastic gradient descent algorithm.

A. U-Net

The first hyper parameter to decide is the learning rate of gradient descent. This controls how much the model changes in response to the estimated error each time the model weights are updated. For the training of the U-Net model, the learning rate used is 0.001. Smaller values of the learning rate have been tested, such as 0.0001, giving very similar results. However, as a smaller learning rate implies a longer training process, 0.001 is the preferred learning rate.

Secondly, one needs to determine the filter number f, which is the number of neurons used in the network. The results obtained for $f = 16, 32, 64$ are presented in the table II, for 2 different size of patches, the next parameter to choose. The results are actually the validation scores obtained when training the model on a 80-20% train-validation split dataset. Three different size of patches have been studied : 16, 128, 256; 16 giving from far the worst results, therefore less investigations have been done and are not presented here. From the table, it is clear that the best choice is to keep a filter number of 32, along with the patch size 256. However, after further investigations, we came to realize that the results are very close for the different filter numbers and the two patch sizes 128 and 256. Since a patch size of 128 obligates us to crop the training images, 256 is the preferred choice.

Patch size \ Filter number	16	32	64
128	0.879	0.878	0.881
256	0.903	0.904	0.903

TABLE II: Validation F1 score for different filter numbers and patches sizes

Finally, the last important hyper parameter to determine is the number of epochs. This one defines the number of times

that the algorithm will go through the entire training dataset. It has to be large enough to minimize the error as much as possible, but not too large to avoid overfitting. An example of a neural network learning is shown in figure 7 in appendix, here the loss function converges in about 5 epoch while the f1 score takes around 100 to 150 epochs to converge and overcome the variations in the validation set. Therefore the chosen number of epochs is 150, so that the f1 score is sure to have converged.

B. Multi-resnet

For the multi-resnet, the hyper parameters used are similar to the ones of the U-Net and have been determined following the same process : learning rate of 0.001, filter number of 32 and patch size of the 256.

For the number of epochs, each independant resnet model is first trained and tested on its own to select the optimal number. The table III presents the F1 score obtained for each model as well as the one obtained for the combination of the three with 70 epochs.

	resnet34	seresnet34	resnet50	Multinet
F1 score	0.873	0.872	0.871	0.875

TABLE III: F1 score obtained for each independant resnet model, and for the resulting multinet

C. Comparison and data augmentation

The table IV summarizes the different parameters selected for each model along with the corresponding F1 scores. For each model, two F1 scores are given : the one obtained without using the augmented training set (only 100 pairs of satellite-groundtruth images), and the one obtained using the augmented training set (1000 pairs of satellite-groundtruth images). The number of epochs are the one used for the augmented set.

	Learning rate	Filter number	Nb of Epochs	Patch size	F1 score
CNN	0.001	64	80	16	0.797 / 0.886
U-Net	0.001	32	80	256	0.864 / 0.910
Multi-resnet	0.001	34,50	150	256	0.881 / 0.913

TABLE IV: Parameters and F1 score obtained for each model

V. Discussion

As shown in table IV, the best score is obtained with the multi-resnet model, followed by the U-Net, with the traditional CNN giving the worst result. The low performance of the CNN compared to the other models is because, while the other models have an encoder/decoder structure specific to image segmentation, the CNN model is a preferred model for object classification. It is not surprising that the ensemble model outperformed the other models, as this method is commonly used in machine learning competitions. The resnet model already outperforms the unet model for f1 scores, therefore in an ensemble model it is sure to be the best model. As expected, data augmentation increased the efficiency of all neural networks, as it allows for better training.

The predicted mask for the two best models are shown in the figure 5 for three different test images. As visible here, both networks are able to predict the location of roads of any orientation accurately. Moreover, one can see that the multi-resnet model is a little bit more precise on the drawing of the border of the roads. In addition, a challenge, which is illustrated in the second aerial image, is when the road is covered by trees. Even though not as accurate, the model is still able to detect the roads.

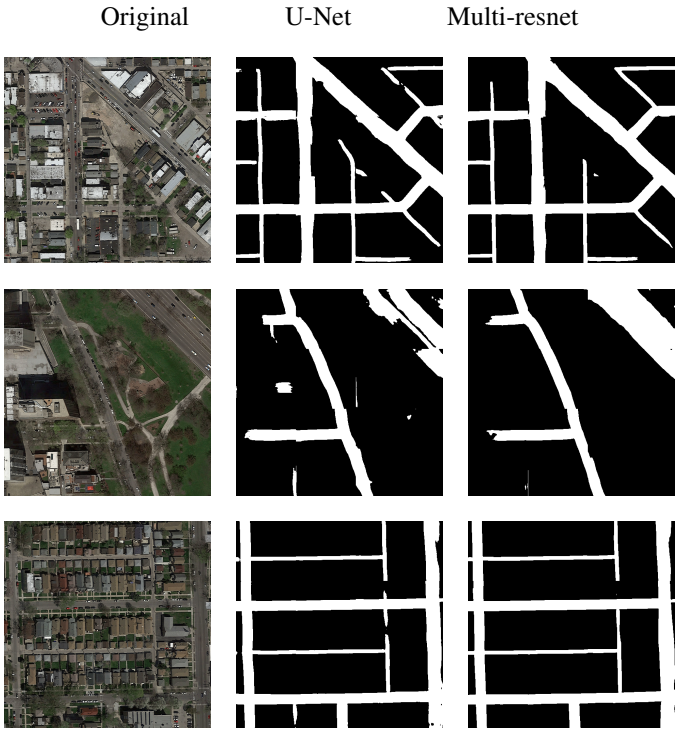


Fig. 5: Predicted groundtruth images for the models implemented

VI. Conclusion

In this project three different models have been implemented, a simple convolutional network, a U-NET and a multi-resnet. The best F1 score: 0.913 has been obtained with the latter. The different investigations have proven that the choice of parameters such as the batch size, the patch size, the number of filters and epochs are important to fine-tune the models to reach a high f1 score. Moreover, an appropriate data augmentation was also a decisive step as it considerably increased the F1 score.

The performance of the network can be improved by increasing the training data through adding more satellite images or with more data augmentation. Moreover using a larger ensemble model with varying high performing segmentation models would also cause an increase in the final f1 score.

References

[1] Lamba, Harshall. "Understanding Semantic Segmentation with UNET." *Medium*, Towards Data Science, 17 Feb. 2019, https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

[2] Brownlee, Jason. "Ensemble Learning Methods for Deep Learning Neural Networks." *Machine Learning Mastery*, 6 Aug. 2019, https://machinelearningmastery.com/ensemble-methods-for-deep-learning-neural-networks/

[3] -, Great Learning Team. "Introduction to Resnet or Residual Network." *Great Learning*, 1 Dec. 2021, https://www.mygreatlearning.com/blog/resnet/

[4] Feng, Vincent. "An Overview of ResNet and Its Variants." *Medium*, Towards Data Science, 17 July 2017, https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

[5] "Papers with Code - Squeeze-and-Excitation Block Explained." *Explained — Papers With Code*, https://paperswithcode.com/method/squeeze-and-excitation-block

[6] "Training on batch: how do you split the data?" https://zerowithdot.com/splitting-to-batches/

[7] "Python Code Examples for generate batches" https://www.programcreek.com/python/?CodeExample=generate+batches

[8] "Effect of batch size on training dynamics." *Medium*, 19th June 2018, https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e

[9] "Convolutional implementation of the sliding window algorithm." *Medium*, 29th June 2020, https://medium.com/ai-quest/convolutional-implementation-of-the-sliding-window-algorithm-db93a49f99a0

[10] "Building a Convolutional Neural Network (CNN) in Keras." *Towards Data Science*, 18th Oct 2018, https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5

[11] "Epoch vs Batch Size vs Iterations." *Towards Data Science*, 23th Sept 2017, https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9

[12] Zhang, Lei, et al. "Figure 1. Splitting an Image into Patches, Where the Whitenbsp;." *ResearchGate*, 4 Oct. 2020, https://www.researchgate.net/figure/Splitting-an-image-into-patches-where-the-white_fig1_2246493049

[13] "Binary Crossentropy Loss Function: Peltarion Platform." *Peltarion* , https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/binary-crossentropy

Appendix

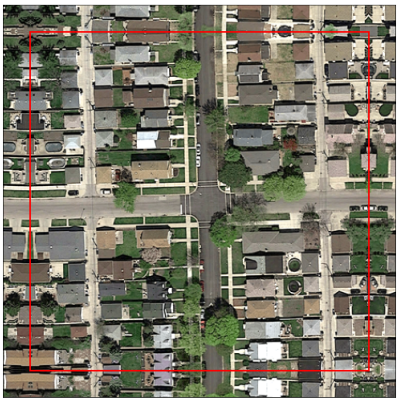
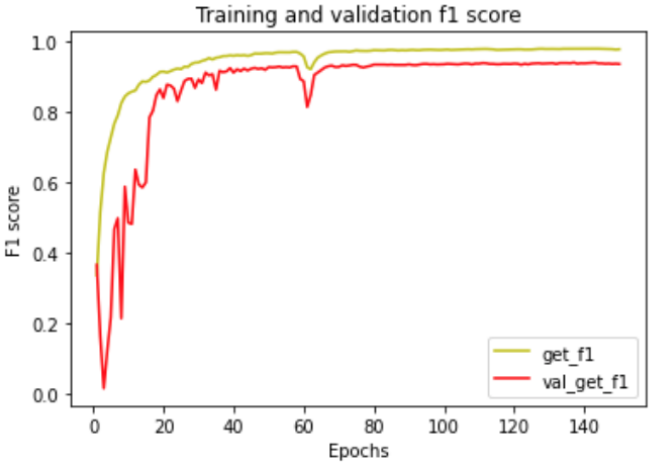
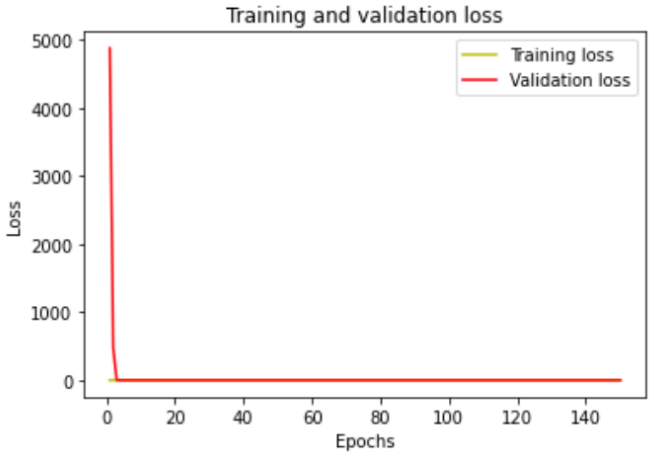


Fig. 6: Image with a 'reflect' padding: the red ligns correspond to the edges of the initial image.



(a) F1 score



(b) Binary cross-entropy loss

Fig. 7: Training of Unet with 256x256 patches and 32 filters