Project 2: Road segmentation

Luc Michels, Grégoire van Oldeneel, Jérôme Lüscher
CS-433 Machine Learning, EPFL, Switzerland

***Abstract*—This project consists in road detection from satellite RGB images. The challenge is to obtain predictions for 50 608×608 test images while having only 100 400×400 images and their groundtruths. A combination of sliding-windows algorithm and an U-Net Neural Network is used to perform the task, providing precise localization and taking into account the context. The dataset is augmented thanks to patches and random transformations in order to make the model robust and invariant to rotations and brightness changes. Predictions are assessed via Aicrowd challenge platform by computing the F1 metric on a binary 16×16 patched image.**

I. Introduction

Pattern recognition and borders detection from images are common challenges for Machine Learning (ML) algorithms. This type of image detection problem is best solved using a Convolutional Neural Network (CNN). O. Ronneberger *et al.* presented in 2015 the U-Net CNN architecture providing great results when applied to biological cells segmentation [1]. We choose to inspire and adapt our architecture from their research since the challenge is quite similar. Indeed, U-Net performs well even with 30 images and our training set is also small. We combine a sliding-window method in order to virtually expand the training set. We also modify these patches with random brightness and rotations to make the net invariant to them. Following sections present our model in depth, as well as the data pre-processing (data augmentation) and the post-processing methods applied in order to obtain the best prediction.

II. Models

During the model selection phase, we explored many possibilities including Logistic Regression, Support Vector Machine, a Simple three layer Neural Network (NN) and an Encode-Decode NN (autoencoder). However, as these models yielded results that were not comparable to those of the other teams and we quickly refocused our efforts on the U-Net CNN.

A. U-Net

O. Ronneberger *et al.* present in [1] a U-Net CNN intended to extract cells borders from transmitted light microscopy images based on a small train set. According to them, the presented architecture outperforms the sliding window convolutional network (as presented by Ciresan *et al.*, [2]). Our idea is here to combine these two methods by first extracting patches from each image and then train a U-Net on all the patches. The U-Net architecture described by O. Ronneberger (illustrated in Figure 1) consists of two branches of 4 layers: contracting first, expansive afterwards. Each layer of the contracting branch contains 2 convolutions (3×3), one Rectified Linear Unit (ReLU) activation layer and a 2×2 max pooling operation which shrinks the output image by half in both x and y dimensions. At each downsampling step, the number of features channel is doubled. In the expansive branch, each layer consists of a 2D transposed convolution layer. This is followed by a concatenation with the cropped corresponding symmetric feature map from the contracting path and finally two 3×3 convolutions. In our implementation, we adapt this architecture with a few noticeable changes. The kernel we use for convolutions are dilated by a factor of 2. This was done in order to better encapsulate the surrounding features because the coverage of a dilated kernel is higher without loss of resolution [3]. We used the `same` padding method instead of the `valid` method in our convolutions to be able to keep the image size constant throughout the two successive convolutions. This means that we could simply concatenate the corresponding images from the contracting and expanding branch without the need to crop. Also, we used a 2D upsampling layer which uses nearest interpolation to fill in the missing values of the resulting image throughout the network followed by 2×2 convolutions for expanding, Whereas a transpose convolution was used in the paper. For kernel initialisation, we first used a normal distribution, as suggested in [1]. Afterwards, we instead used the uniform Glorot distribution described in [4] which yielded better results.

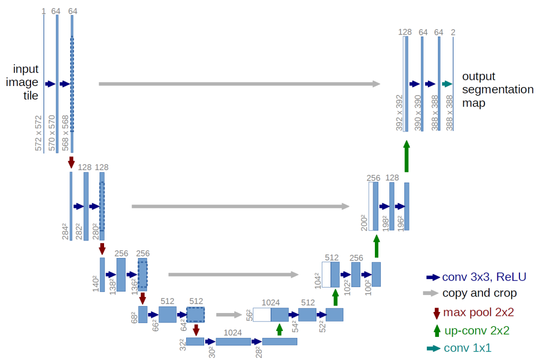

Figure 1. U-Net CNN Architecture presented in [1].

1) Optimisation Algorithm: We tried the classical Stochastic Gradient Descent (SGD) algorithm as suggested in [1], however we found the Adam optimizer performed much better in this task. The Adam optimizer is an improved version of the SGD algorithm and requires less tuning of the parameter values. Indeed, the Adam optimiser uses individual learning rates for different parameters of the network based on the first and second moment of the derivatives. We used the parameters proposed in the paper [5], namely a starting learning rate of 0.01 for all parameters.

2) Padding: The `same` zero-padding method allowed us to keep the size of the image constant. This differs from the technique used in the paper where only valid features were used in order to only use meaningful feature information. However, when rotations of random angles were applied to the data, it was imperative to apply another padding method in order to avoid having black areas at the edge of the image and thus lose information. For this case, we chose mirror padding.

3) Number of layers: We implemented two different architectures with 4 and 5 layers respectively. The maximum number of channels increases from 16×the number of input layers to 32×the number of input layers and the minimum size of the image in the network also decreases by half. We notice that 5 layers U-Net does not provide better results (see Table I) and took much longer to train because it contains around four times more parameters.

4) Loss function: We chose Binary Cross Entropy Loss (BCE) as it strongly penalises wrong classifications:

$$loss = -\sum_{i=1}^{m} y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)}) \quad (1)$$

with m, the number of pixels, $y^{(i)} \in \{0, 1\}$, the true result and $\hat{p}^{(i)} \in [0, 1]$, the estimation obtained. We also implemented a weighted BCE loss function that yields a higher value for the false negatives because of the over representation of negative examples (non-roads) in the dataset. However, this variant did not yield better predictions and was thus discarded.

B. Data Preprocessing

1) Patches: We speculated that the network might perform better if the 400×400 images were split into smaller overlapping patches. This has four clear advantages. First, the test images which have a size of 608×608 can also be split into patches of the same size as those of the training set. Second, the smaller size might allow the network to detect smaller features and thus be more precise by fully utilising the information contained in the images. Third, the overlapping patches provides more data to train on and multiple predictions for a single image area. These predictions can then be aggregated in a meaningful way in the post-processing step. Finally, the network could learn translation invariance because we use a stride of 16 and the patches overlap $(128 - 16)$ columns or rows of pixels. In essence, the next patch can be seen a translation of the previous one. We found that this technique greatly improved the predictions on the validation set, as seen in Table I)

2) Image transformations: O. Ronneberger *et al.* [1] present the case of training a CNN with only a few samples. Data augmentation with image transformations allows the CNN to become invariant to these transformations when they appear in the test set. As their purpose was cell border recognition, they choose to apply shift and rotation invariance and robustness to deformation and brightness. We found the following transformations to be relevant for our task: random rotations in a given range, horizontal and vertical flipping, zoom and brightness variations. We justify this choice by analysing the image data. It appeared that most of the images in the test set contain either horizontal or vertical roads, but the test set contains many images with diagonal roads. We chose to zoom the images as it the test images that were slightly larger may have been taken in a different resolution. The random changes in brightness are important because the images may look vastly different depending on the time of the day where they were taken . Also, shadows were obscuring roads and made predictions harder for the network (e.g. as shown in Figure 2).

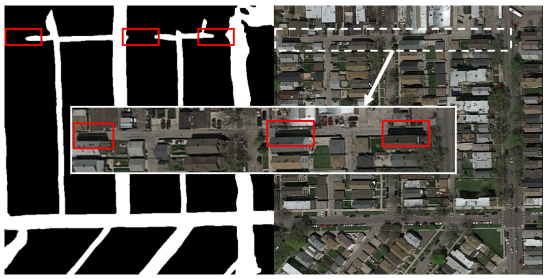
Figure 2. Prediction before brightness invariance data augmentation on test image 7. Roads shaded by buildings are undetected.

We found that our networks were able to overfit on both the original and the augmented dataset, but they generalised better by training on the augmented dataset.
We did not include shifts in our transformations because cropping the images into overlapping patches already makes them shift invariant.

C. Data Post-processing

1) Aggregation of predictions: The test images were split into patches with a stride = 8, which means that there is a significant overlap. Figure 3 shows the number of overlapping layers. For each pixel we computed the mean value of all the overlapped predicted patches. Then we use a threshold of 0.25 as it was used in the benchmark algorithm provided in order to produce a binary output image.

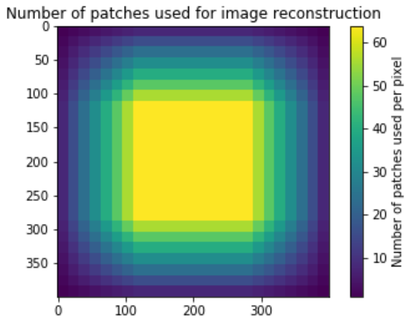

Figure 3. Heatmap showing the number of overlapping patches used to compute the output image. Patch size = 128, stride = 16 pixels.

2) Removal of small objects: Taking a look at our reconstructed predictions on the test set images, we noticed small objects in between roads (pink spots in Figure 4), so we implemented a method for removal of blobs of connected pixels below a fixed threshold. The chosen threshold was a rectangle size of 32×16 pixels, which means that we remove objects that are smaller than two 16×16 submission patches.

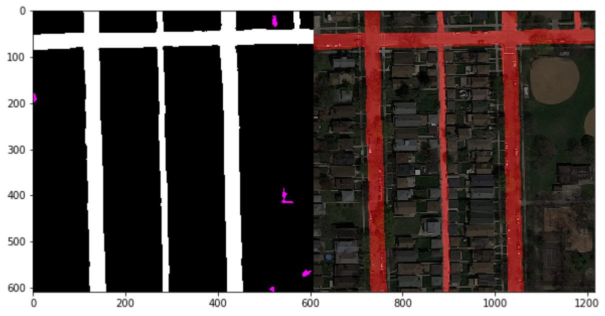

Figure 4. Left: prediction result on test image 29 where objects to remove are shown in pink. Right: Overlay of test image with our prediction after removal of objects. We see in this particular instance that only non-road predictions are removed.

D. Overfit

Overfitting on the training dataset was the main challenge that we had to overcome during this project, on almost all of the early loss plots, a very evident gap between the validation and the test loss was present. The following approaches were implemented to deal with this problem. We implemented a 80-20 training/validation split on the data in order to be able to test the prediction performance of our NN locally and also to avoid overfitting by comparing the F1 scores on the train and validation dataset.

1) Dropout: The dropout consists of randomly selecting a fixed percentage (0.2 in our case) of nodes in the previous layer. These are disabled for the computation of the values of the next layer. The literature indicates that the addition of a dropout layer after the application of a non-linear activation can help reduce overfitting [6]. Dropout allows to reduce the number of never activated neurons and thus obtain a similar mean activation value for each neuron. Our dropout probability was chosen according to the publication where the minimum classification error for a CNN on the MNIST dataset was found using a dropout with a probability of 0.2 [6].

2) Early stopping: A key aspect to prevent overfitting is to avoid to train the network for too long, we thus monitored the F1 score on the validation dataset and we halted the training algorithm if the validation loss value had not improved after arbitrary amount of epochs, called 'patience'. We also reduced the learning rate by a factor of 80% once the validation loss stopped improving with a patience of 5 epochs.
We found our validation F1 score to rise very rapidly at the beginning, but then to reach a plateau after approximately 10 epochs (as seen in Figure 6), so it was also critical to implement a reduction to the learning rate in order to avoid oscillating around local minimas.

III. Libraries used

The code use the *Keras* framework was run on the *Google Colab* GPU for better performance.

IV. Submission pipeline

We present a summary of our methodology for the best performance. We first split the sample set into 80 training samples and 20 validation samples. Each sample is then sliced according to a sliding-window method (window 128×128, stride = 16) combining context and precision. In order to augment the train set, at each epoch, all the training windows are randomly transformed (rotation range of 45^o, filled with mirror padding). The network is a 4 layers deep U-Net CNN, as presented before. The learning rate is defined by Adam algorithm [5]. Although, if a plateau is detected, the learning rate is reduced by a factor of 80%. The number of epochs is not limited (max = 500 which is never reached). The test set is sliced into a larger amount of windows as we believe that overlap may increase the likelihood of the result thanks to the aggregations. Therefore we choose 128×128 patches with a stride of 8 pixels. After predicting each patch, a prediction image is reconstructed by taking the mean value of overlapped pixels. A binary image is obtained with a 0.3 threshold. This value was determined empirically. Post-processing is used in order to clean the prediction test images. Small disconnected spots are removed since they are not considered as roads. Finally,

Model	Random transformation	F1_tr	F1_val	F1_sub
UNet 4 l.	None, no patch	0.962	0.796	/
UNet 4 l.	None	0.971	0.973	0.868
UNet 4 l.	h.v. flip	0.961	0.963	0.876
UNet 4 l.	45^o rot	0.914	0.942	0.871
UNet 4 l.	45^o rot, h.v. flip	0.868	0.886	0.864
UNet 4 l.	45^o rot, zoom 0.9	0.905	0.930	/
UNet 4 l.	45^o rot, bright, cutout	0.612	0.838	/
UNet 4 l.	45^o rot, bright, h.v. flip	0.938	0.942	**0.908**
UNet **5** l.	45^o rot	0.911	0.924	0.904

Table I
SUMMARY OF THE MAIN OBTAINED RESULTS. L. = LAYERS; PATCH BY DEFAULT: 128×128, STRIDE=16; ROT = ROTATIONS; H.V. = HORIZONTAL/VERTICAL; BRIGHT = BRIGHTNESS; _VAL = VALIDATION; _TR = TRAIN; _SUB = SUBMISSION.

images are exported and transformed into `.csv` files for submission.

V. Results

The main results (presented in Table I) were obtained using similar method as the one described in Section IV.

The loss and F1 score evolution are shown in Figure 5 and 6 for our best model (U-Net CNN, rotation and brightness invariance, 4 layers, dilation = 2, dropout = 0.2), presented in Section IV. We notice that the F1 score for the validation set is higher that the training F1 score. This might seem odd but it is explained by the fact that the random transformations were only applied to the train set, as the validation set is only used for testing. Results have been improved step by step thanks to adequate data augmentation based on observations on the test images. It is interesting to notice that local F1 results differs from submission results. We claim that it is due to differences between the two sets. For instance the higher proportion of diagonal roads in the test set. One interesting observation is that combining random rotations and horizontal/vertical (h.v.) flip decreases the result. The last training augmentation we added is random brightness since we believe that shadows can limit the detection. This last addition allows us to reach our best result. The F1 score on the test set for the final model is **0.908** and was obtained using the exact submission pipeline described in Section IV.

VI. Discussion

We found that rotation and brightness data augmentations were able to improve the training allowing us to reach satisfying results. Nevertheless, we believe that further improvements can be made.
In the test images, some roads are hidden by trees, buildings, bridges etc. T. Devries *et al.* suggest a *cutout* method to overcome this challenge described in [7]. It consists in adding random obstacles to force the model to ”guess” road positions hidden behind it. We tried to launch this method but the training was too slow (9h) and the F1_val, while increasing, was still low. Another possible improvement could be to apply individual weights to each sample proportional to the amount of non-roads to penalize false negatives more. This takes into account the larger examples of non-road compared to road pixels. Another improvement would be to include mirror padding into the U-Net, as suggested in [1].

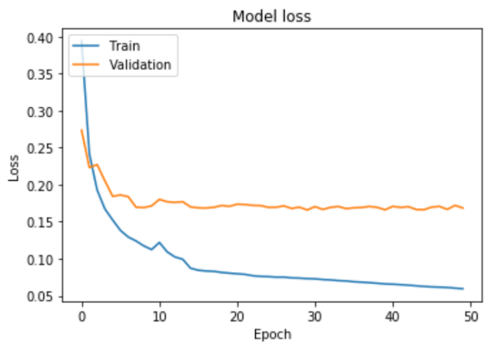

Figure 5. Plot of the training and validation loss against training epochs.

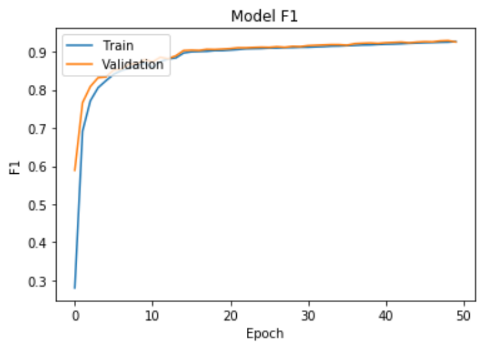

Figure 6. Plot of the training and validation F1 score against training epochs

VII. Conclusion

By using an 4 layers depth U-Net CNN architecture inspired and adapted from the literature, made robust thanks to training data augmentation and post-processing, the model is able to detect the roads with high fidelity and a F1 score of 0.908 on the AIcrowd contest. Some improvements are still possible like a cutout method in order to augment the train dataset and a mirror-padding in the convolution layers. In summary, the improvements obtained by augmenting the dataset may provide better results but require more computing power and take longer to train. This trade-off is always to be taken into account when selecting the data augmentation to perform.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[2] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 2843–2851.

[3] R. Hamaguchi, A. Fujita, K. Nemoto, T. Imaizumi, and S. Hikosaka, "Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery," *CoRR*, vol. abs/1709.00179, 2017. [Online]. Available: http://arxiv.org/abs/1709.00179

[4] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html

[5] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[6] S. Park and N. Kwak, "Analysis on the dropout effect in convolutional neural networks," in *Computer Vision – ACCV 2016*, S.-H. Lai, V. Lepetit, K. Nishino, and Y. Sato, Eds. Cham: Springer International Publishing, 2017, pp. 189–204.

[7] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *CoRR*, vol. abs/1708.04552, 2017. [Online]. Available: http://arxiv.org/abs/1708.04552