Road Segmentation with Convolutional Neural Networks

Capucine Berger-Sigrist, Mihai David, Tiberiu Mosnoi
Department of Computer Science, EPFL, Switzerland

***Abstract*—In this project we exploit the power of deep learning on a semantic segmentation problem, identifying roads in images. We use the U-net architecture as the backbone of our approach and employ various Machine Learning techniques such as transfer learning, data augmentation and hyper-parameter tuning in order to push the model to its limit. We also present how understanding and exploiting the underlying structure of this particular problem was absolutely critical in obtaining a strong learner which achieved 0.917 F1-score on the test set.**

I. Introduction

From pathology detection in medical research to self-driving cars, Convolutional Neural Networks have proven to be very efficient when it comes to computer vision related tasks. More specifically, they can, with certain architectural modifications, be applied to obtain an accurate semantic segmentation. Hence, given a set of aerial images from GoogleMaps, our aim is to train a neural network to segment roads by assigning a label to each pixel of these images. As our research shows, it can be done fairly accurately, even with a restricted amount of data, using transfer learning and clever data augmentation techniques.

II. Dataset

The dataset we were provided with is composed of 100 satellite images with their associated ground-truth. Each image of the training set is 400x400 pixels, while the images of the test set have a higher resolution of 608x608 pixels. As for the ground-truth, we were given probabilistic labels, rather than binary labels, such that:

$$label = \begin{cases} 1 & \text{i.e road, if } p_{label} > 0.5 \\ 0 & \text{i.e background, otherwise} \end{cases}$$

where p_{label} is the given probabilistic label.

III. Network architecture

Semantic segmentation requires that a label be assigned to each pixel. Classical convolutional networks, while well suited to simple classification tasks, lose information about the localisation of the label during the process of prediction. As such we needed a different model, which is where the U-Net architecture from Ronneberger et al. [1] came into play.

A. U-Net

The U-Net model uses an Encoder-Decoder architecture: the task of the decoder is to semantically project the discriminative features (lower resolution) learnt by the encoder onto the pixel space (higher resolution) to get a dense classification.

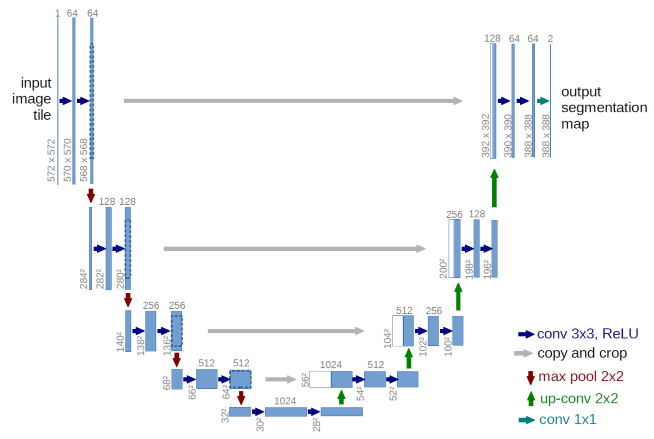


Figure 1: U-Net architecture from Ronneberger et al.

The U-Net architecture (Figure 1, [2]) consists of two branches, a down-sampling branch and an up-sampling branch. The down-sampling branch is a sequence of convolutional and max-pooling layers and its purpose is to extract the most important features of the data. Then, the up-sampling branch uses those features to construct the semantic map of the original image.

The up-sampling branch can have different flavours. For instance, in the original architecture, this branch uses transposed convolutions to convert back the image to its original resolution. However, we decided to use an alternative architecture that relies on bi-linear up-sampling because it has fewer weights to train and consequently the computational cost of learning is smaller and it also implicitly guards against over-fitting (because the complexity of the model is reduced).

B. Transfer Learning: VGG-13

For the down-sampling branch, we decided to use the weights from VGG-13 [3], a pre-trained model on ImageNet. In order to perfectly match the pre-trained weights with our U-Net architecture, we changed the number of channels in the last feature extraction block from 1024 to 512. Thus, all the up-sampling blocks have their number of channels halved compared to the original architecture. The advantage of this approach is twofold: the performance of our model improved and the training time decreased. First, the quality and relevance of the extracted features is top-notch since VGG-13 was trained on a very large and broad dataset and its results were impressive. It was of no surprise to us that our model performed better (as it can be seen in Section II) when we used those weights instead of training them ourselves. Also, now we had fewer weights to train so the training time improved considerably.

IV. Implementation details

In this section we will see in more detail how our model was trained.

A. Loss function

We used a combination of the classic binary cross-entropy loss (1) and the dice loss functions (2):

$$CE(y,p) = -y * log(p) - (1-y) * log(1-p) \quad (1)$$

$$DL(y,p) = 2 * \frac{\sum_{n=1}^{N} p_n * y_n}{\sum_{n=1}^{N} p_n^2 + \sum_{n=1}^{N} y_n^2} \quad (2)$$

where p is the softmax probability, and y the class label.

Taking the sum of both losses enabled us to benefit both from the increased gradient stability with the cross entropy-based component, and the robustness to class imbalance from the Dice-based component, as explained in [4].

We also experimented with the focal loss but it brought no improvements (see Section VII for more details).

B. Hyperparameters

We chose the RMS_prop optimization algorithm (with a step decay update policy for the learning rate) because it tends to converge faster to the optima than other optimizers, in particular for non-convex problems [5]. An advantage of using an adaptive learning rate optimizer was that if the model stopped improving for a certain amount of epochs, the learning rate would be automatically reduced in order to search for the local minimum with more fine-grained steps.

Table I presents the hyperparameters that were used for all the different models.

LR Factor	Patience	Learning rate	Momentum
1e-1	3	1e-4	0.9

Table I: Hyperparameters.

C. Dropout

To resolve the problem of over-fitting encountered during various stages of training, we introduced a dropout [6], which randomly deactivates some nodes in the network. This dropout was added after each max pooling layer during down-sampling with a probability of 0.25 for the first layer and 0.5 otherwise, and then again after each concatenation (i.e. before going to the next convolution layer) in the up-sampling branch, also with a probability of 0.5. This regularization technique forces the model to rely on more robust features and also to not become too complex.

D. Evaluation metrics

To evaluate the prediction of the validation set against the ground-truth, we used the Dice score (F1-score), which tracks the prediction per pixel and is defined as:

$$DiceScore = 2 * \frac{overlap\_area}{total\_number\_of\_pixels\_in\_both\_images}$$

V. Experimental setup

In this section, we will explore how we preprocessed the images in order to improve the robustness of the classifier.

A. Data augmentation

Given the relatively small size of our dataset, we applied data augmentations to our training set, which presented two major advantages. The first one is that by teaching our model to be invariant to slight perturbations, it consequently became more robust. Moreover, this also enabled us to introduce more variability in our dataset without needing more data [7]. During the project we experimented with different augmentations: horizontal and vertical flips, rotations of the whole image and also brightness, contrast and gamma adjustments.

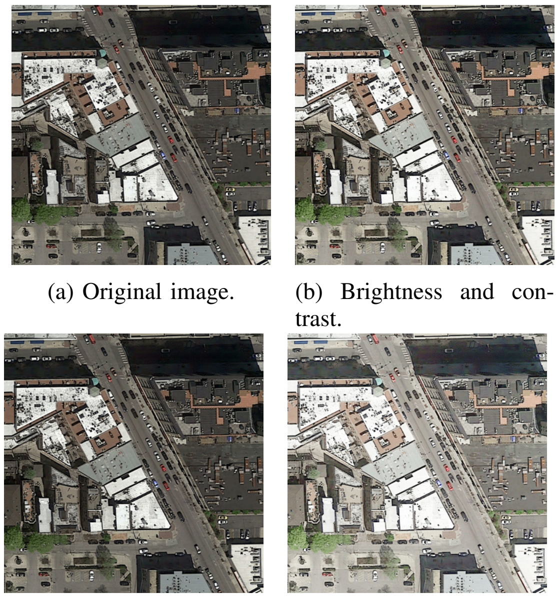
1) Rotations: Apart from some rare images of curved roads in the training set, we found that the majority were either horizontal or vertical straight lines. This could prove problematic, as our model might be confronted with pictures of roads at a a different angle and would then not be able to correctly segment it. To make our model invariant to these kinds of modifications (e.g. a road at a 45° angle), we applied a set of rotations of [45°, 90°, 135°, 180°, 225°, 270°, 315°] to each image.

2) Pixel adjustments: Given the fact that the training set came from real life, it was expected that not all images would have the same lightning conditions. It is reasonable to believe that our model had a harder time classifying some of the darker pixels (which could belong to roads that were in the shadow of some building, for example).

To ameliorate this we applied a linear transformation to each pixel:

$$pixel = contrast * pixel + brightness$$

where the brightness and contrast were randomly selected from the intervals [-50, 50] and [0.5, 1.5] respectively.



(a) Original image. (b) Brightness and contrast.

(c) Original image. (d) Gamma correction.

Figure 2: Example of pixel adjustment to reduce the effect of the shadows from the buildings. Applying either a brightness and contrast correction or a gamma correction to the original image.

Finally, we added a gamma filter, which helps with under or over-exposed images. We used a set of correction such that $\gamma \in [0.6, 0.8, 1.2, 1.4]$, where a $\gamma < 1$ will typically darken an image, whereas $\gamma > 1$ will brighten it. Figure 2 shows an example of how pixel adjustments can be used to improve the original image and thus the prediction output by the model.

B. Test-time augmentation

In order to gain the most out of the data augmentation techniques we had used, we decided to also employ test-time augmentation [3]. Concretely, for each test image we constructed several other images by performing horizontal and vertical flips and rotations with [90°, 180°, 270°] on the original image. Then we used our model in order to extract the semantic maps of these augmented images and finally we aggregated them into one final semantic map (Figure 3). The aggregation operation is pixel-wise, i.e. for a pixel we find all corresponding pixels' probabilities in the semantic maps of the augmented images and then apply the aggregation operation on those. We tried both averaging and taking the maximum and concluded that averaging works slightly better.

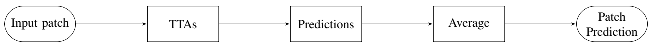


Figure 3: Test-time augmentation pipeline.

C. Test set preprocessing

One important challenge that we were faced with was how to handle the different resolutions of the images in the training set (400x400) versus the ones in the test set (608x608). It turns out that finding a satisfactory solution to this impediment was crucial to obtain a competitive score which goes to show how important it is to adapt and tweak Machine Learning techniques to a given problem as dictated by its unique structure.

Our first idea on how to handle the fact that the images from the test set had a higher resolution than in the training set was to resize them to 400x400 pixels. Then we would apply our model in order to extract the semantic map and resize it to its original resolution of 608x608 pixels using mirror padding.

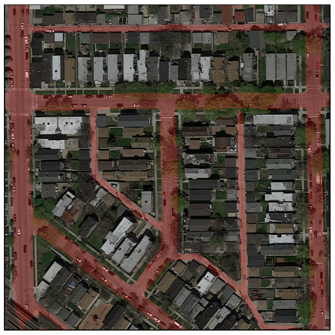
However, a more satisfactory approach was to split the 608x608 image into four 400x400 images, each containing one of the four corners, run our model on them and then construct the semantic map of the original 608x608 image by aggregating the four smaller overlapping semantic maps. We tried both averaging and taking the maximum as aggregation strategies and concluded that averaging yielded slightly better results. This approach mode more sense because we no longer had to downscale the test images, and waste precious information in the process, in order to pass them to our network.

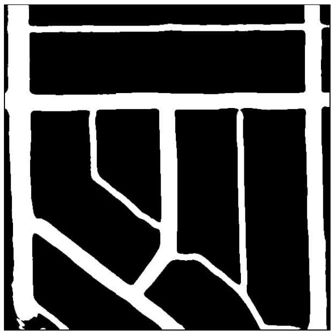
VI. Results

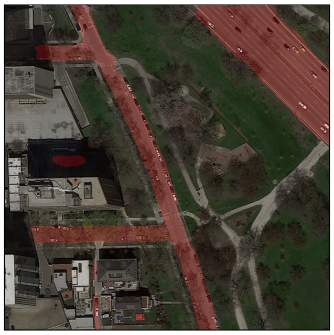
We started with the modified version of the U-Net architecture (as explained in Section III) and trained it on the training set of images of resolution of 400x400. In order to be able to use this trained model on the 608x608 test images, we down-scaled them, applied our model and then up-scaled the result (**RSZ**). Then we modified our model to use the weights from a pre-trained model (**PTW**) which yielded a considerable improvement of **9%**. Adding data augmentation (**DA**) resulted in an another **2%** improvement. The quantum leap came when we changed our strategy (as explained in Section V-C) for dealing with the difference in resolution of the training and test images (**PTCHS**). This combined with the addition of dropout (**DRPO**) yielded an impressive **8%** boost in the score. We then refined our model by adding test-time augmentations (**TTA**) and another kind of data augmentation, gamma correction (**DAG**), which pushed the score another **0.5%** further. Seeing as our model still over-fitted, we decided to do a grid search for the weight-decay hyperparameter (Table III) and this yielded an additional **0.2%** increase.

VII. Discussion

In our process of data exploration, we noticed that our dataset is slightly imbalanced: the ratio of road to background pixels is 1:4. We thought that this bias could negatively affect the performance of our model so we decided


(a) Overlap n°10.


(b) Prediction n°10.


(c) Overlap n°26.

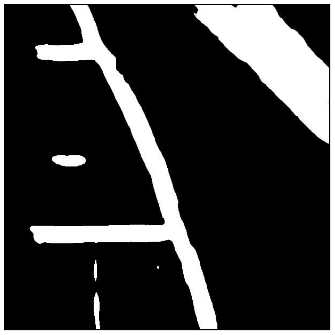

(d) Prediction n°26.

Figure 4: Segmentation obtained for image n°10 and n°26 from the test set.

Model	F1-score
Baseline : U-Net + RSZ	72.5%
U-Net + RSZ + PTW	81.6%
U-Net + RSZ + PTW+ DA	83.3%
U-Net + PTCHS + PTW + DA + DRPO	91.0%
U-Net + PTCHS + PTW + DAG + DRPO + TTA	**91.7%**

Table II: Results comparison between the different models. **RSZ**: Test set images resized to 400x400 pixels. **PTW**: Pre-trained weights from VGG-13. **PTCHS**: Test set images cropped into four patches of 400x400 pixels. **DA**: Data augmentation. **DRPO**: Dropout. **TTA**: Test-time augmentation. **DAG**: Data augmentation with gamma correction.

Weight decay	F1-score
1e-8	91.5%
1e-7	91.6%
1e-6	**91.7%**
1e-5	91.5%

Table III: Grid search results for the weight decay.

to try to ameliorate it by using the focal loss function [8] instead of the classical cross entropy loss.

The focal loss for binary classification is defined as follows:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise} \end{cases} \quad (3)$$

where $y \in \{\pm 1\}$ is the ground-truth and p the estimated probability of the label $y = 1$. And then from (3) we obtain:

$$FL(p_t) = -\alpha(1 - p_t)^{\gamma} log(p_t)$$

with $\gamma = 2$ and $\alpha = 0.25$.

Using the Focal loss did not yield better results however, probably because a skewness of 1:4 is not pronounced enough for it to be an impediment for our model.

VIII. Summary

The backbone of our approach consists in a modified version of the U-Net architecture. Relying on transfer learning to train part of our model, we then enriched the dataset using data and test-time augmentations, thus further improving its robustness. The real breakthrough came when we changed the way the test set was preprocessed: precisely, when we moved from the rescaling approach to splitting the 608x608 test image in four 400x400 images, computing their semantic maps and aggregating them to obtain the 608x608 final semantic map.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015, version: 1. [Online]. Available: http://arxiv.org/abs/1505.04597

[2] H. Lamba. Understanding semantic segmentation with UNET. [Online]. Available: https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition." [Online]. Available: http://arxiv.org/abs/1409.1556

[4] M. Yeung, E. Sala, C.-B. Schönlieb, and L. Rundo, "Unified focal loss: Generalising dice and cross entropy-based losses to handle class imbalanced medical image segmentation." [Online]. Available: http://arxiv.org/abs/2102.04525

[5] V. Bushaev. Understanding RMSprop — faster neural network learning. [Online]. Available: https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[7] J. Nelson. Image augmentations for aerial datasets. [Online]. Available: https://blog.roboflow.com/image-augmentations-for-aerial-datasets/

[8] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection." [Online]. Available: http://arxiv.org/abs/1708.02002