Project 2 - Road Segmentation Challenge

Emilio Fernández, Tyler Benkley, Frédéric Berdoz
Machine Learning, EPFL, 2019

***Abstract*—In the context of image segmentation, this paper explores three different machine learning approaches in order to perform road segmentation on a training set of 100 satellite images and their respective groundtruths. The task is first attempted with logistic regression coupled with feature engineering. Then a convolutional neural network (CNN) is applied to cropped patches of the image, and finally more complex and specific CNN inspired by U-nets to the whole image. On the basis of a standard F1 score on a test set of 50 images, the U-net resulted in the best performance showing the power of deep learning techniques in image segmentation.**

I. Introduction

An important task in image processing with applications in computer vision is image segmentation [1], which consists of classifying each pixel of an image according to what it represents. In this project, the goal was to apply image segmentation on satellite images acquired from Google Maps, using different machine learning algorithms. More precisely, the task was to train a classifier that could identify whether a pixel was a road or not. In addition to the satellite images, corresponding groundtruth images were provided such that the classifier could be trained using supervised learning techniques.

The following report is structured as follows: First, in sections II and III, an exploratory data analysis and data preprocessing are performed with the aim of better understanding the data and extracting its most essential characteristics. In section IV, three different models are presented in order to compare their performances. Firstly, a regression model was trained, and since the task at hand was binary classification, regularized logistic regression was used (RLR). Classical models have the advantage of being relatively cheap in terms of computational resources. However, for better performance, they require fine-tuned feature engineering which we will discuss later. Secondly, a classic convolutional neural network (CNN) was developed. CNNs are computationally expensive but are known to perform well in image segmentation. Lastly, a more complex type of neural network, known as a U-NET, was trained. It is basically a more advanced CNN that performs not only down-sampling but also up-sampling to achieve a better segmentation (see Fig. 3).

At the end of this paper, sections V and VI present the performance of the different models as well as ideas on how to further improve them.

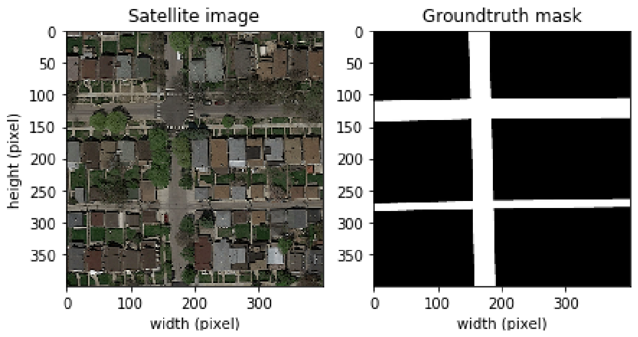


Figure 1: Satellite image (left) and groundtruth (right)

II. Exploratory Analysis

The training dataset consisted of 100 satellite images and 100 corresponding groundtruth masks. Each data point was a 400x400 satellite image in RGB format (see Fig. 1).

The groundtruth masks are in grey level format. In the masks, road pixels are white (1) and background pixels are black (0), which corresponds to the prediction of the classifier. For computational and contextual reasons, the images and the masks were divided into patches of 16 by 16 pixels. A patch is considered to represent a road if more than 25% of its pixels represents a road. With this data processing, 74.656% of the training patches are background and 25.344% are roads, which means that a deterministic classifier would have a 74.656% accuracy on the training data. As for the testing dataset, it consisted of 50 608x608 satellite images, also in RGB format. Although different in size, the scales of the training and testing images appeared to be the same.

III. Data Preprocessing

Data preprocessing and feature engineering play an important role in machine learning tasks. For this project, the data was processed at three different levels:

- at the pixel level (*Channel augmentation* or CA),
- at the patch level (*Feature extraction* or FE),
- at the image level (*Data augmentation* or DA).

A. Channel Augmentation

Channel augmentation consists of adding channels to each pixel. We added the following channels: a grey level channel (CA$_g$), a vertical edge channel (CA$_v$) and a horizontal edge channel (CA$_h$). The last two channels were obtained by convolving 3×3 kernels over the images, respectively ω_v and ω_h:

$$\omega_v = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad \omega_h = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

The images were mirrored at the edges in order to avoid any discontinuity while convolving the kernels.

Channel augmentation is important in regression methods since it extracts information that the model would not have detected otherwise. However, for CNNs, an adequate model autonomously finds the kernel that extracts the best information. Therefore, it makes little sense to use these data processing techniques for deep learning models.

B. *Feature Extraction*

For the RLR, since the input of the machine learning model is a collection of patches and not of pixels, one must extract features from all the pixels constituting a patch. In order to do so, we simply computed the mean (FE^m) and standard deviation (FE^{sd}) of each channel over each patch. In addition, we built a new feature matrix consisting of all the monomials of degree at most ϕ in the d original features (FE_ϕ). For instance, if $x = [x_1\, x_2]^\top$ was a data point, the augmented data point $\tilde{x} = \text{FE}_2(x)$ would be given by:

$$\tilde{x} = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 \end{bmatrix}^\top.$$

As previously explained for the channel augmentation, feature extraction was only required for classical models such as our RLR model.

C. *Data Augmentation*

Since the number of training images was relatively small, augmenting the training data boosted the performance of the model. We decided to rotate each image and its groundtruth by a given angle α (DA_α), and also applied vertical symmetry (DA^s). In order to deal with the fact that a rotated square image leaves the corner of the original frame empty, reflection padding was employed for filling these corners.

For the U-Net, reflection padding was also used to extend the training images such that their size would match that of the test images. This was required since the input of the U-Net model was the full image, whereas the inputs of RLR and CNN models were patches. Reshaping either the test or the train data would have introduced a bias since the scale correspondence would have been lost.

The data augmentation here was very beneficial to our neural networks. As shown in Table III, the advanced models boasted a significant increase in performance after augmenting the training set.

IV. Models and Methods

As mentioned in section II, accuracy (i.e. the proportion of patches that are correctly classified) is not a good measure of the model performance. A better way of evaluating the precision of the classifier is to use the F_1-score, which is defined as follows [2]:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

This will be the default metric to compare the performance of th three following models.

A. *Regularized Logistic Regression*

In order to optimize the regularization term λ of the logistic regression model, k-fold cross-validation was performed within our computational limits. Table III presents the performance of each model and its corresponding data preprocessing, with RLR_c standing for a regularization term of $\lambda = 10^{-c}$.

B. *Convolutional Neural Network*

As for the neural network approach, various models were attempted on multiple types of inputs. After much trial and error, the model that performed best was a relatively simple pyramidal CNN (see Table I). It involved taking a patch with three channels (RGB) as input and outputting the probability of that patch being a part of the background and the probability of the patch being a road. It was noticed that including the context of the patch (i.e including a strip of p pixels of its surrounding, denoted by C_p in Table III) significantly improved the performance. In our final model, we decided to use a 3x3 patch grid as the input of our CNN (i.e. C_{16}). Hence, the input size was 48 x 48 pixels.

Table I: CNN architecture

Layers	**Parameters**
Input	48 x 48 x 3
Conv2D, ReLu	64 layers (7, 7), p = 0.25
Conv2D, ReLu	64 layers (5, 5), p = 0.25
Conv2D, ReLu MaxPool	64 layers (3, 3) (2, 2)
Conv2D, ReLu	128 layers (3, 3), p = 0.25
Conv2D, ReLu MaxPool	128 layers (3, 3) (2, 2)
Conv2D, ReLu MaxPool	256 layers (3, 3) (2, 2)
Dense, ReLu	128, p = 0.25
Dense, ReLu	256, p = 0.5
softmax, output	2 x 1

p: Dropout rate

Using Google Colab [3], we were able to train our model over 100 epochs using 125'000 patches. Fig. 2 illustrates the training history of the model, where the validation test accuracy is computed using 20% of the total number of patches. One can see that over-fitting occurs even though regularization was introduced via the dropout rates. Further increasing these rates resulted in decreasing the convergence time and was therefore avoided.

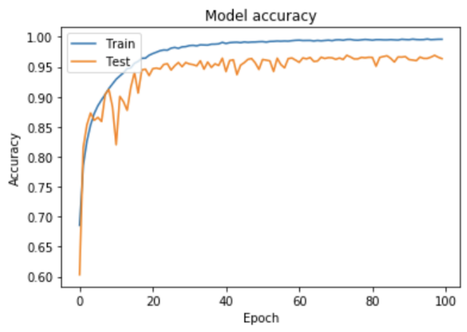


Figure 2: Training history of the CNN model.

C. U-Net

The model that resulted in the best performance was inspired by the U-net introduced in [4] whose full architecture is presented in Table II. It is a more complex convolutional neural network consisting of a contracting path followed by a symmetrically expanding path that allows precise binary segmentation (see Fig. 3). In addition, some convolutional layers that are in the contraction path are copied in the augmentation path. Such neural networks are known to perform notably well on small datasets that are augmented using extensive data augmentation [5], which is the case in this project. As seen in Table III, data augmentation (by a factor of 18) sensitively increased the performance of the model. The final model was trained using the binary cross-entropy loss function. The training dataset was composed of 1620 images and the validation dataset of 180 images. After 200 epochs, the training process reached a state in which the validation loss was not decreasing anymore. Further increasing the number of epoch would not have improve the model performance.

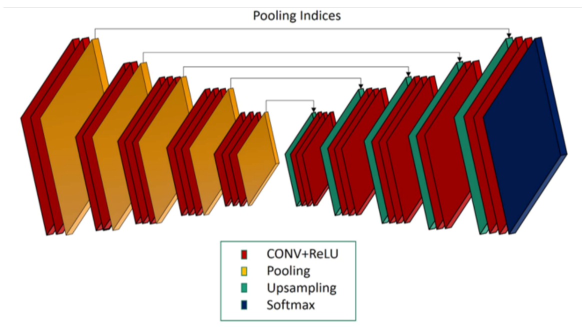


Figure 3: U-net architecture. (Figure adapted from [6])

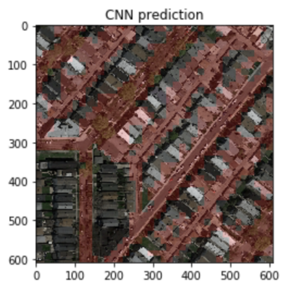
Table II: U-Net architecture

id	Layers	Parameters
i	Input	608 x 608 x 3
c1	Conv2D, ReLu Conv2D, ReLu	16 layers (3, 3), p = 0.1 16 layers (3, 3)
p1	MaxPool	(2, 2)
c2	Conv2D, ReLu Conv2D, ReLu	32 layers (3, 3), p = 0.1 32 layers (3, 3)
p2	MaxPool	(2, 2)
c3	Conv2D, ReLu Conv2D, ReLu	64 layers (3, 3), p = 0.2 64 layers (3, 3)
p3	MaxPool	(2, 2)
c4	Conv2D, ReLu Conv2D, ReLu	128 layers (3, 3), p = 0.2 128 layers (3, 3)
p4	MaxPool	(2, 2)
c5	Conv2D, ReLu Conv2D, ReLu	256 layers (3, 3), p = 0.3 256 layers (3, 3)
u1	ConvDTrans + c4	128 layers (2, 2), stride (2, 2)
c6	Conv2D, ReLu Conv2D, ReLu	128 layers (3, 3), p = 0.2 128 layers (3, 3)
u2	ConvDTrans + c3	64 layers (2, 2), stride (2, 2)
c7	Conv2D, ReLu Conv2D, ReLu	64 layers (3, 3), p = 0.2 64 layers (3, 3)
u3	ConvDTrans + c2	32 layers (2, 2), stride (2, 2)
c8	Conv2D, ReLu Conv2D, ReLu	32 layers (3, 3), p = 0.1 32 layers (3, 3)
u4	ConvDTrans + c1	16 layers (2, 2), stride (2, 2)
c9	Conv2D, ReLu Conv2D, ReLu	16 layers (3, 3), p = 0.1 16 layers (3, 3)
o	Conv2D, sigmoid Output	1 layer (1, 1) 608 x 608

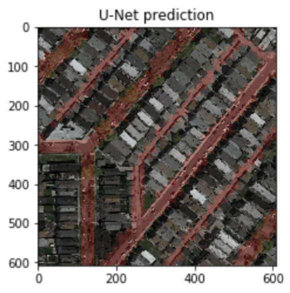
p: Dropout

V. Results

Table III presents the performance of all the models that were developed in this project. In addition, Fig. 4 compares the prediction of the CNN model (Fig. 4a) and the U-Net model (Fig. 4b). It can be see that the CNN model tends to overestimate the size of the road whereas the U-Net model is really accurate.



(a) CNN prediction



(b) U-Net prediction

Figure 4: Comparison between CNN and U-Net models.

Table III: Model Performance

Model	**Data Preprocessing**	**F1** [%]	
		Train	Test
RLR_{10}	$\text{FE}_3^{m,sd}$	55.7	**51.5**
RLR_{10}	$\text{CA}_{g,v,h}$ + $\text{FE}_3^{m,sd}$	56.1	**53.1**
RLR_{17}	$\text{DA}_{0,45,90}$ + $\text{FE}_3^{m,sd}$	60.8	**55.9**
RLR_{17}	$\text{DA}_{0,45,90}$ + $\text{CA}_{g,v,h}$ + $\text{FE}_3^{m,sd}$	64.8	**61.0**
CNN	-	74.5	**74.9**
CNN	$\text{DA}_{0,45}$	78.1	**75.9**
CNN	$\text{DA}_{0,45}$ + C_{16}	81.7	**78.6**
U-Net	-	89.3	**88.8**
U-Net	$\text{DA}^s_{0,45,90}$	90.7	**89.4**
U-Net	$\text{DA}^s_{0,30,45,60,90,120,135,150,180}$	92.3	**90.0**

VI. Discussion

Our results state that a neural network is more effective than a RLR model. This is the case because, in regression models, the input is an individual patch characterized by its features. In spite of possessing additional information in terms of channels, it is completely detached from its original image. Spatial correlation in the image is therefore lost and the model is not able to exploit this essential component. This sets the ceiling for any individual patch classifier. However, with a neural network, the model convolves over the entire image and benefits from the spatial correlation of the pixels. The caveat is that it must learn the importance of this spatial correlation for every pixel with the others in the image. Hence, it requires an abundant dataset with many different configurations of roads. Furthermore, since our inputs are entire images, we suffer a reduction in the number of data points, but paradoxically, we require more. This is the main trade-off in using a neural network. Anyhow, through data augmentation, we were able to expand our data set and surmount this problem. Thus, our neural networks exceeded the performance of RLR. In order to further improve the performance of our model, on could for example optimize the regularization to minimize over-fitting. One could also try to train two different models and then combine their predictionor to develop a more advanced neural network.

VII. Summary

In the end, we were able to train a U-Net model that can segment an image into road and background with an F_1-score of 90%. We could compare this model with a classical regression model (RLR) and with a simpler CNN model. Although more accurate, the U-Net required more computational power and more data for the training process. This trade-off is at the heart of all machine (and deep) learning tasks. On the other hand, we noticed the data preprocessing and feature engineering is generally less intensive for neural network since, in a way, the model autonomously finds the best features.

All in all, we were satisfied with the general accuracy of our model and the predicted masks were not too far from what our brains can actually predict (as shown in Fig. 4b).

References

[1] S. Yuheng and Y. Hao, “Image segmentation algorithms overview,” *CoRR*, vol. abs/1707.02051, 2017. [Online]. Available: http://arxiv.org/abs/1707.02051

[2] Wikipedia contributors, “F1 score — Wikipedia, the free encyclopedia,” 2019, [Online; accessed 4-December-2019]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=F1_score&oldid=928547822

[3] Google, “Google colab — Google,” https://colab.research.google.com/notebooks/welcome.ipynb, accessed: 2019-12-18.

[4] Kaggle contributors, “Keras u-net starter,” https://www.kaggle.com/keegil/keras-u-net-starter-lb-0-277, accessed: 2019-12-18.

[5] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[6] K. Saleh, M. Hossny, and S. Nahavandi, “Kangaroo vehicle collision detection using deep semantic segmentation convolutional neural network,” *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1–7, 2016.