EPFL Machine Learning Project 2 Report
Road Segmentation
Team: The Overfitters

Dario Pavllo*, Mattia Martinelli† and Chanhee Hwang‡
École polytechnique fédérale de Lausanne
Switzerland
*dario.pavllo@epfl.ch, †mattia.martinelli@epfl.ch, ‡chanhee.hwang@epfl.ch

***Abstract*—This project aims to build a classifier that segments satellite images, where the segmentation part consists in separating the areas that represent roads from the rest. Among various techniques, only convolutional neural networks (CNNs) have been shown to solve this task with satisfying results. Specifically, our method is based on a *sliding window* approach, and has been deployed on a CNN that makes use of Leaky ReLU activation functions and Dropout layers. Moreover, real-time image augmentation has proved to improve the accuracy of the model and avoid overfitting. According to our experiments, this model outperforms all the other tested techniques.**

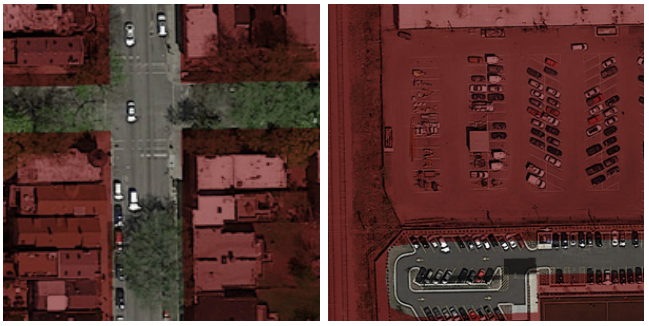
I. Introduction

Image segmentation is a technique that is becoming increasingly popular for various tasks in computer vision. Generally speaking, this process consists in labelling every part of an image according to certain criteria. For instance, such a technique could be used for face detection in photos, or detection of roads in autonomous driving vehicles.

Recently, the increase of computing performance, as well as the ability to exploit massively parallel computation with GPUs, has led to the development of new machine learning techniques that are able to process images in reasonable time. However, image processing has always been a challenging task, as the information is organized in a definite geometrical structure, and therefore algorithms should consider their morphology. In addition, the computational cost to process an image does not scale linearly with its size, and this has led to the development of new techniques such as *convolutional neural networks*, which foster sparse connections and weight sharing in order to reduce the complexity of the problem.

The aim of this project is to build a model that is able to perform the segmentation of satellite images. Specifically, the segmentation consists in detecting which parts of the images are roads, and which parts are background (e.g. buildings, fields, water).

This report provides a brief overview of different methods that can be used to solve this problem, and particularly it addresses convolutional neural networks, which represent the state-of-the-art technique for image classification. Specifically, Section III proposes several approaches and methods that can be used to perform this task. The rest is organized as follows: Section IV provides some implementation details that have proved useful to improve the performance of our model; Section V explains the methodology of the accuracy validation of the proposed techniques, and Section VI gives a comparison of the obtained results.



(a) Trees over roads (b) Parking lot

Fig. 1: Some areas from the training set and their respective ground truth masks (in red).

II. Exploratory data analysis

The dataset consists of 100 satellite images of urban areas and their respective ground truth masks, where white pixels represent roads (foreground) and black pixels represent the rest (background).

The task is to classify blocks of 16×16 pixels, considering that the label associated with each block corresponds to 1 if the average value of the ground truth pixels in that block is greater than a threshold (0.25), 0 otherwise.

By looking at the training set, it can be observed that the classification task is not trivial, as some roads are covered by trees. Furthermore, some asphalt areas are not labelled as road (e.g. parking lots and walkways), and this could potentially confuse the training model. Figure 1 shows these complications.

For these reasons, it is agreeable to think that the classifier should take into consideration a sort of context, i.e. it should look at nearby pixels in order to infer some information about the block that is being classified.

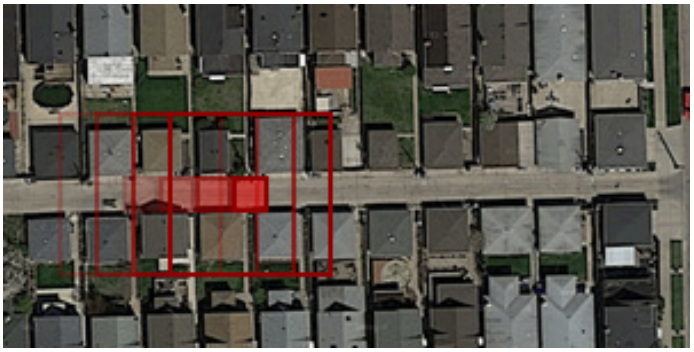
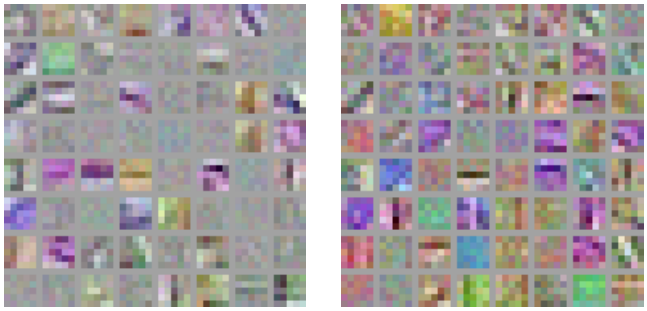


Fig. 2: Sliding window approach. The small square at the centre is the patch (of size 16×16) that is being classified, whereas the big square represents the current context (i.e. *window*, of size 72×72). In this figure, subsequent windows are spaced apart by 16 pixels (*stride*).



(a) Dead filters (ReLU) (b) Good filters (Leaky ReLU)

Fig. 3: Visualization of the filters in the first layer, with the same model and the same training set, but different activation functions.

III. Models and methods

In order to the evaluate the quality of the proposed model, it is important to have a baseline model that will be used as a comparison for the classification accuracy. Based on the observation that there are fewer foreground areas than background areas, the first baseline model that has been used classifies all blocks as background (i.e. 0).

It is possible to improve the initial result by using a linear classification model, such as the logistic regression. However, for the reasons mentioned in the previous section, such a model would not be able to correctly process the context of images, since it would not be capable to detect their morphological structure (unless complex feature extraction techniques are used).

In order to confirm this claim, a logistic regression model has been implemented for comparison purposes. The input features correspond to the mean (over the entire 16×16 patch) and the standard deviation of each RGB channel (for a total of 6 features), and they are transformed according to a polynomial basis of degree 4 (including interactions as well).

However, the most reasonable choice would be to use a model based on *convolutional neural networks* (CNNs), since they are well suited for images. Indeed, this model has provided excellent results on our dataset and has been adopted as final solution.

According to our research, several methods have been proposed to solve the task of per-pixel classification; the one adopted in this project consists of a sliding window approach [1]: the objective is to classify the block at the centre of an image, according a certain context, which in this case corresponds to a square window of size `window_size` $\times$ `window_size` (a hyperparameter). Figure 2 shows this technique more clearly.

For what concerns the neural network structure, the number of layers and filters has been optimized to perform well on this dataset. Furthermore, the following features have been explored:

A. Activation functions

ReLUs are the standard choice for deep neural networks; however, when a high learning rate is used, some units can get stuck and cause the so-called *dead filters*. This problem can be mitigated by using a lower learning rate, at the cost of a longer training time (which can be already excessive). For this reason, a variant known as *Leaky ReLU* has been used as the activation function for all intermediate layers, with good results. It is defined as $f(x) = \max(\alpha x, x)$, with $\alpha \ll 1$, and in our case α has been chosen to be equal to 0.1 . Although this might seem a high value, some studies have shown that higher values perform better than lower ones [2], and with this dataset $\alpha = 0.1$ has proved effective to prevent dead filters, as shown in Figure 3.

B. Image augmentation

Since the dataset is very small (100 images), an image augmentation strategy has been adopted to virtually increase its size. Specifically, before being supplied to the neural network, each training sample (i.e window) is randomly rotated in steps of 90 degrees, and it is also randomly flipped horizontally/vertically. This effectively yields an increase of the dataset size by a multiplicative factor of 8, and has been shown to greatly improve the accuracy of the model. The implementation details of this technique are shown in section Section IV.

C. Regularization

Although the dataset augmentation helps to reduce overfitting, the use of *Dropout* layers has been very effective in our model. They have been added after each max-pooling layer (with $p = 0.25$), and also after the fully connected layer (with $p = 0.5$). Furthermore, L2 regularization has been used for the weights (and not biases) of the fully connected and output layers, with $\lambda = 10^{-6}$.

The window size has been empirically chosen in order to take into account a context that is large enough, considering that large windows are computationally expensive. Therefore,

Type	Notes
Input	$72 \times 72 \times 3$
Convolution + Leaky ReLU	64 5×5 filters
Max Pooling	2×2
Dropout	$p = 0.25$
Convolution + Leaky ReLU	128 3×3 filters
Max Pooling	2×2
Dropout	$p = 0.25$
Convolution + Leaky ReLU	256 3×3 filters
Max Pooling	2×2
Dropout	$p = 0.25$
Convolution + Leaky ReLU	256 3×3 filters
Max Pooling	2×2
Dropout	$p = 0.25$
Fully connected + Leaky ReLU	128 neurons
Dropout	$p = 0.5$
Output + Softmax	2 neurons

TABLE I: Full list of layers in the neural network.

a size of 72×72 has proved to be a good compromise. Table I shows the complete structure of the proposed neural network, which is the result of various experiments.

IV. Implementation Details and Training

The neural network has been implemented and trained through the *Keras* library, which is well suited for prototyping and can use either TensorFlow or Theano as backend.

One problem with per-pixel classifications is that a *stride* (i.e. how much the sliding window moves between each training sample) has to be chosen. Ideally, to guarantee that the model is shift-invariant, the stride should be equal to 1; however, this leads to an extremely large training set which does not fit in memory, and the condition is worsened by the dataset augmentation. As a consequence, we have implemented a real-time training set generator that works as follows: at each iteration, the algorithm generates a mini-batch composed of randomly sampled windows from the original training set; all images are transformed according to the image augmentation technique described in Section III-B, and their ground truth is calculated; finally, they are supplied to the learning algorithm. This method ensures that, eventually, every possible area of the training set will be explored. Furthermore, the algorithm is executed on a separate thread, thereby causing no performance hit.

For what concerns the behaviour at image borders, mirror boundary conditions have been applied, i.e. the image is reflected along the boundary axis, as shown in Figure 4. This produces a good estimation, especially because the majority of images are vertically or horizontally aligned.

The model has been trained with a NVIDIA GeForce GTX 960 graphics card (with 2 GB of VRAM), using 32 bit floating-point precision and Theano backend. The loss function is the *softmax categorical cross-entropy*, which has been minimized using the Adam optimizer [3] (a variant of SGD), with automatic learning rate adjustment once the accuracy reaches a plateau for a certain number of iterations (it is halved when the training accuracy has not improved over the last 10 iterations). Furthermore, the initial learning rate is set to $\eta = 0.001$ and the batch size is 125 samples.

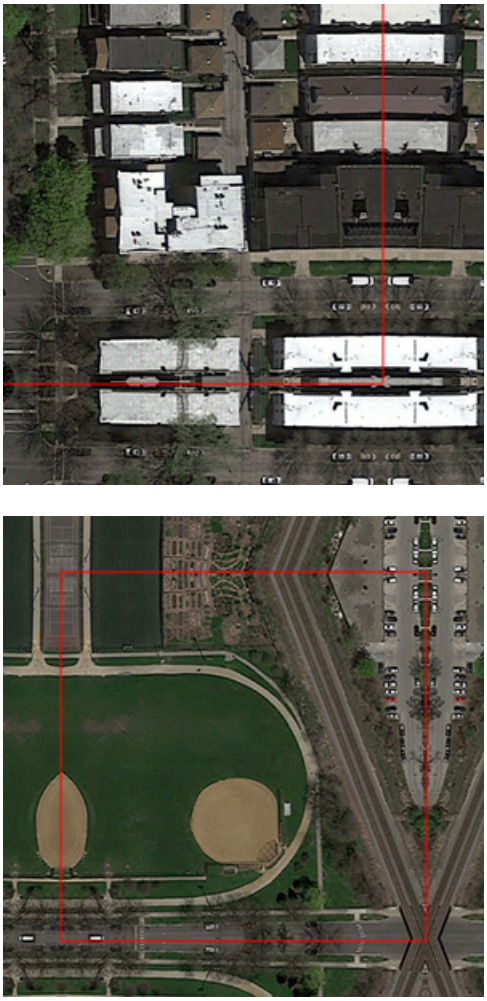


Fig. 4: Mirror boundary conditions. The red line has been added for illustration purposes, and shows the original boundaries of the images.

V. Validation of Results

In order to verify the robustness and the accuracy of the model, it must be tested with a known validation set.

As a first approach, the predictions have been tested with a static validation set. The original training dataset has been separated into two subsets: 75% of the dataset has been used to actually train the model (*training set*), while the remaining 25% has been used to perform cross-validation (*validation set*). This simple method, which corresponds to a partial 4-fold cross-validation, does not require long execution cycles to be executed, and therefore it has been preferred for the first experiments. After having selected a number of candidate models, they have been validated with a full k-fold cross-validation algorithm (with $k = 4$), which has been used

to estimate their final accuracy, along with their standard deviation.

Since convolutional neural networks require a long time to be trained, it is not feasible to optimize their hyperparameters through a grid search. Therefore, they have all been tuned manually according to various heuristics.

VI. Final results

The final cross-validation results are shown in Table II and Figure 5. As can be observed, the approach based on the convolutional neural network with Dropout layers, Leaky ReLU activations and image augmentation is clearly superior, with an accuracy of $92.89\% \pm 0.7\%$. Furthermore, this value is expected to be greater once the model is trained with the full dataset.

The difference between the methods based on CNNs is small, but still significant.

#	Model	Accuracy $\pm\sigma$
A	All background	$74.09\% \pm 1.2\%$
B	Logistic regression	$78.53\% \pm 0.2\%$
C	CNN	$92.05\% \pm 0.9\%$
D	CNN + LR	$92.22\% \pm 0.9\%$
E	CNN + LR + D	$92.57\% \pm 1.0\%$
F	**CNN + LR + D + IA**	$\mathbf{92.89\% \pm 0.7\%}$

TABLE II: Tested models along with their respective cross-validation results.
Legend: CNN: Convolutional Neural Network; LR: Leaky ReLU; D: Dropout; IA: Image Augmentation.

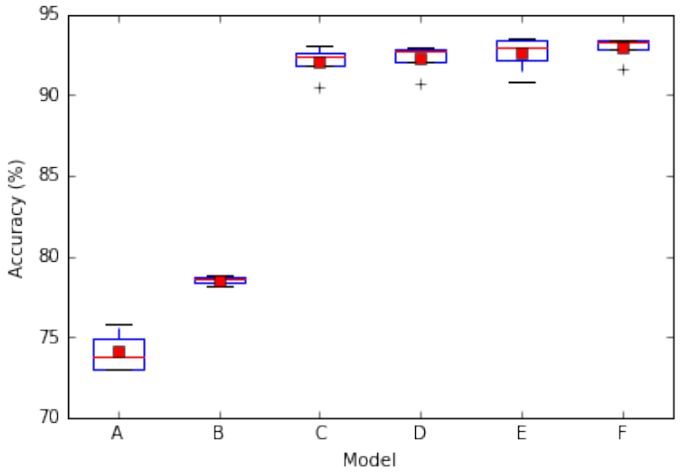


Fig. 5: Comparison among the different models.

VII. Conclusions

In conclusion, the performance of the final model has proved to be satisfactory. As can be observed in Figure 6, which shows some examples of classification on the test set, the neural network segments the image in a human-like manner. In particular, the results seem to be correct even in border cases such as streets covered with trees or image boundaries. However, the model also tends to produce glitches in certain cases, e.g. diagonal streets. This might be caused by the fact that block-classification (in 16×16 patches) is not suitable for these cases, as well as the fact that the majority of the training set is aligned either horizontally or vertically.

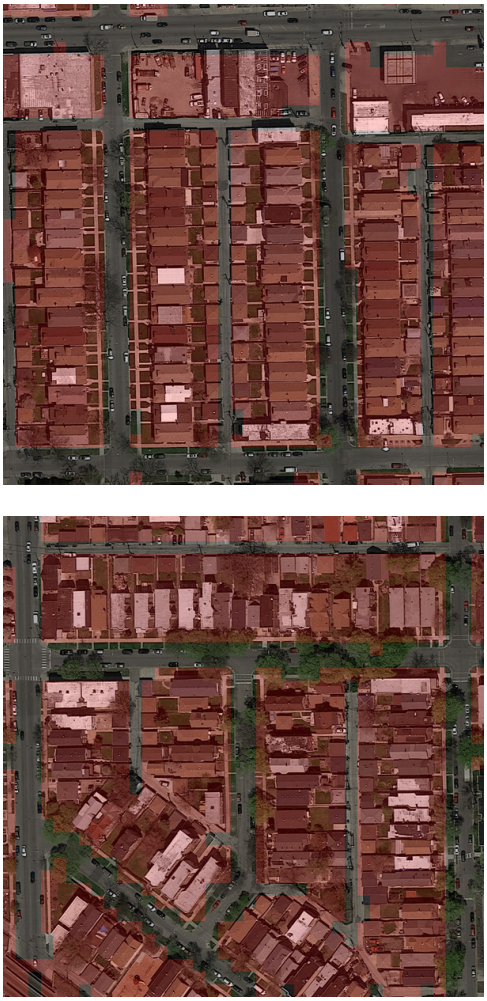


Fig. 6: Segmentation of two images from the test set.

Nevertheless, the final result is quite impressive, and is comparable to human performance.

References

[1] S. Bittel, V. Kaiser, M. Teichmann, and M. Thoma, "Pixel-wise segmentation of street with neural networks," *arXiv preprint arXiv:1511.00513*, 2015.

[2] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[3] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.