Machine Learning Project 2 Report
Road segmentation

Quentin Rebjock
Msc in DS, EPFL Lausanne

Bojana Ranković
Msc in CS, EPFL Lausanne

***Abstract*—Image classification is a hard machine learning problem whose resolution techniques have progressed rapidly in recent years. The road segmentation consists in assigning the correct label, either 'background' or 'road', to each pixel of satellite images. Modern approaches to this question such as convolutional neural networks were investigated. It turned out that image augmentation improved the results significantly, especially for diagonal roads which are rare in the original training set.**

I. Introduction

This project aims at developing a machine learning model capable of achieving the segmentation of pictures taken from a satellite. More precisely, given a set of ground images on which are present buildings, roads, trees or rivers, we would like to distinguish which pixels correspond to the roads and which are just the background. The following figure shows what kind of images had to be classified and how it had to be performed.

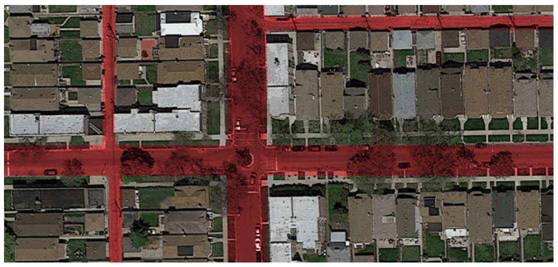

Fig. 1. Segmentation of a satellite image. The red overlay separates the roads and the background.

It is a question of classifying images and assigning them the right label. Image classification is a tough task and some machine learning techniques are more suitable than others in this area. In particular, basic models like the logistic regression are very unlikely to give satisfying results and, on the contrary, convolutional neural networks proved to be very adapted to that kind of function.

This report describes the problem rigorously, exposes what methods can be considered, how they were implemented and the performances that resulted.

We are first formalizing the problem, giving more details about what exactly is expected, presenting the models that were explored in the section III, and finally elaborating the implementation in the part IV and analyzing the results in the section V.

II. Problem formalization and exploratory data analysis

The final goal is to classify each 16x16 tile of a set of 50 images in one of the following two categories : either background, corresponding to the label 0, or road which is associated to the label 1.

A training set of 100 satellite images and their matching ground truth black and white images is provided, each of them being 400x400 pixels size. Obviously, the decomposition into 16x16 patches is inaccurate : the same 16x16 tile may contain a road and a building at the same time. Since the task is to classify these small segments, each ground truth image from the training set has to be preprocessed in such a way that one single class (0 or 1) is assigned to each 16x16 tile. The criterion that was chosen is to consider a tile to be a road if the mean value of its pixels is greater than a certain threshold, namely 0.25 in this case. It is important to note that around 73% of the image is the background and there is only 27% of roads ; the labels are very unequally distributed.

A naive approach consisting to train the models using only these 16x16 patches is very likely to fail because of the complexity of the roads. For instance, a road might be hidden by some trees and it will be labeled as the background if it's not taking into account the context near the considered tile. The car parks are also tough to classify because they look like normal roads but are in fact labeled as background. Once again, the context around the tile might help the model to find out the right label.

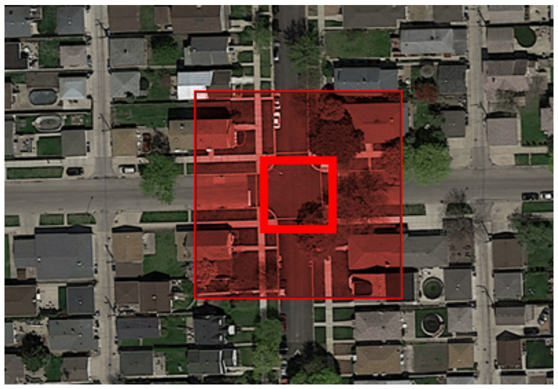

Fig. 2. Around the 16x16 patch, a larger window whose size can be tuned is classified so the context around the tile is considered

The figure 2 shows how to extend the 16x16 patches in such a way that the pixels around it are also examined. The window size is a parameter that can be tuned and we expect larger values to give better results (but also costing more computation time). For the tiles close to the borders, the image can be padded using a symmetric filling.

III. Investigated models

Several types of models can be set up for this classification task and two main categories were implemented and tested : the baseline models like the logistic regression or the support vector machines, and the advanced models using the convolutional neural networks (CNN). But first, notice that a completely naive model predicting 0 (background) for every single tile gives roughly 73% accuracy because of the repartition of the labels, as stated above. Let's detail more the baseline and the advanced models.

A. Baseline models

We obviously can't expect very good results from the baseline models but it is important to have an idea of their performances and to see if they give satisfactory classifications. All these models were trained with 16x16 patches and their corresponding labels, without taking the context around it. Since these models need a feature matrix to work, we need to extract relevant features from each patch.

1) Logistic regression: The logistic regression is the simplest model that was trained for this project.

2) Support vector machines: A standard support vector machines classifier using the radial basis function kernel was trained. Only the regularization parameter had to be tuned.

3) Random forest: The random forest classifier was validated through cross-validation with only 10 estimators and then used with 500 estimators. Indeed, the performances of such a classifier usually increase with the number of estimator (that parameter being unlikely to overfit).

None of these models gave really satisfying results. That is why we moved to more complex models, namely neural networks.

B. Convolutional neural networks

The convolutional neural networks are likely to give way better results than the baseline models presented above. They are indeed known for their great accuracy in image classification tasks. For these models, the context of each patch was taken in consideration as explained above. A window whose size can be tuned and centered on the patch is classified, with the same label as the original tile. A significant advantage of these models compared to the baseline models is that the features are automatically learnt, and we don't have to guess what kind of features play an important role.

1) Sequential networks: The first CNN that we tried to implement were sequential networks composed of several two dimensional convolutions, pooling, dropout and dense layers.

2) U net: The U net architecture is more complex than a simple sequential network. One of their important specificities is to have concatenation bridges between several layers of the network. They are also supposed to be very reliable for image segmentation tasks. The following figure shows what a U net looks like.

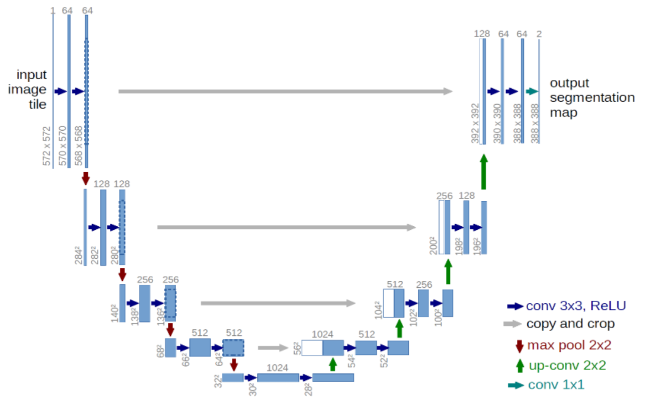

Fig. 3. Representation of a UNet architecture

3) Regularization: Convolutional neural network are usually not prone to overfit a lot and several regularization methods exist. Dropout layers can help to significantly improve the results by reducing the overfitting and performing model averaging. A standard L2 regularization can also be added and both techniques were experimented.

IV. Implementation, training and validation details

To implement these models, the python libraries scikit-learn and Keras (under TensorFlow backend) were used. They provide reliable and effective functions to develop machine learning models, to train them, and to validate them.

A. Baseline models

In regards to the baseline models, we chose to extract two basic features, the mean and the variance of each patch, which seemed relevant to do the classification. These features were then enhanced using a polynomial basis of degree 5 to try and improve the accuracy. The baseline models hyperparameters are few so a simple grid-search coupled to cross-validation is enough to tune them.

B.

Neural networks

As for neural networks, several architectures and hyperparameters were tested because there is no general rule stating what structure is the most adapted to the classification task. A lot of parameters can be tuned : the filter size (in the convolutions), the number of layers, the dropout rates, the regularization parameters, the activation functions, and so on.

The optimization algorithm that was chosen is the Adam optimizer, which is popular in the field of deep learning for achieving good results quickly. Since the classification is done in two categories, the binary cross-entropy loss function was a natural choice.

Also, it turns out that having a deeper network with smaller filter size is often better than a shallow network with larger filter size. CNN with 2, 3, 4, 6, and 8 2D convolutions were trained and it seemed like the accuracy was increasing with the number of convolutions, with a peak at 6. Then the network started to overfit and larger number of convolutions were not attempted.

The first convolution's filter size was always 5x5 and then 3x3 in the deeper layers. Once again, these values were found empirically because there is no rigorous way to find the optimal value.

Two activation functions were inspected : the classic ReLu and the variant leaky ReLu which allows a small non-zero gradient instead of being equal to 0 when the unit is not active. Using this variant leaky ReLu improved the results very slightly.

The training of the CNN is extremely long on common laptops. For this purpose, we used two Titan X Maxwell (12 Go) GPU which made the computations much faster. Since the score used on the competition website Kaggle.com for this project was the micro averaged F1-score, all the cross-validations were made in respect to that specificity.

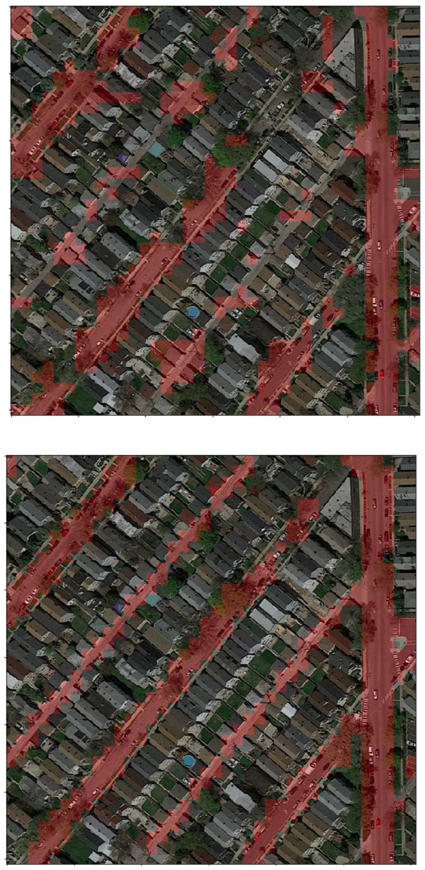
Fig. 4. Two predictions from the exact same model, but without and with image augmentation respectively

C. Image augmentation

The data set being composed of only 100 images (or 62,500 patches) It prevents overfitting on the one hand and more data means better accuracy for most machine learning algorithms on the other hand. Enhancing the set of images in this case is pretty easy because all the rotations of a multiple of 90° and all the symmetries generate new ready to use images. However, it appears that most of the roads in the training data set are either horizontal or vertical and it causes diagonal roads to be often misclassified. Thus, rotations of 45° (or even better : rotations of any degree) would help to enhance the data set in a very efficient way but these are a little more tricky than the rotations of 90° . We cannot simply rotate the images because they would not be rectangular anymore, and the labels change with the rotation. The library Keras includes a preprocessing module which helped to overcome these problems. The provided image augmentation gives any kind of rotation and symmetry.

The following figure shows how the image augmentation improves the performences. The two predictions were made with the exact same model, but the first one without all the rotations, and the second one with all of them. It clearly helps for diagonal roads.

D. Window size tuning

As stated above, the window size is a hyperparameter that can be tuned to improve the results. A larger window size means more context taken in consideration around the patches but also a more complex model and more training time. To validate that parameter, we trained the same models with several window sizes and found out that the accuracy increases with the window size until a certain threshold, in the range $[60, 80]$. Most of our neural networks were thus trained with window sizes of 64 or 72 pixels.

V. Results

The results and the performances of the different models that we trained are presented in this part.

Obviously, the baseline models don't give great accuracy scores, as expected. The naive model consisting in predicting 'background' for all the tiles gives a score of 73% and serves as a reference.

The following table exposes the results of the cross-validation on the relevant experimented models. All the neural networks are here using a window size of 72, leaky ReLu activation functions and a L2 regularizer.

Index	Model	Accuracy
1	Naive	0.734 ± 0.018
2	Logistic	0.734 ± 0.018
3	SVM	0.779 ± 0.027
4	Random forest	0.782 ± 0.017
5	4 convolutions not augmented	0.925 ± 0.035
6	4 convolutions augmented	0.928 ± 0.033
7	**6 convolutions augmented**	0.946 ± 0.031
8	U Net	0.915 ± 0.0223

Surprisingly, the results of the logistic regression are in no way better than the naive model : it predicts background everywhere.

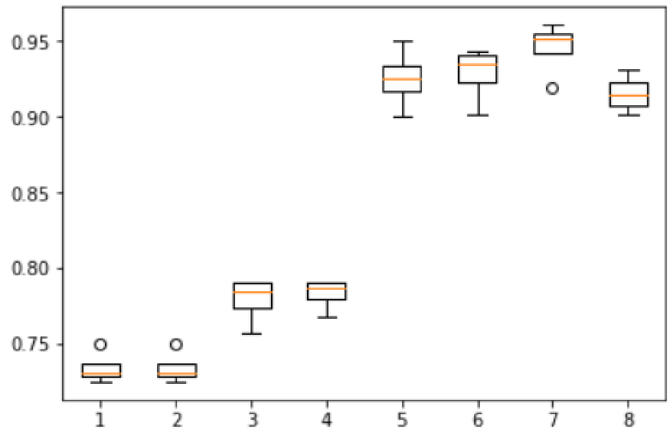

Fig. 5. Box plot comparison between the different models

Finally, our best model led us to get a score of 0.94337 on the platform Kaggle.

VI. Conclusions

As a conclusion, our final best model gives very satisfactory predictions, although not perfect. The paths hard to classify such as train roads, car parks, or roads covered by trees are overall well managed. The following figure shows two examples of predicted images.

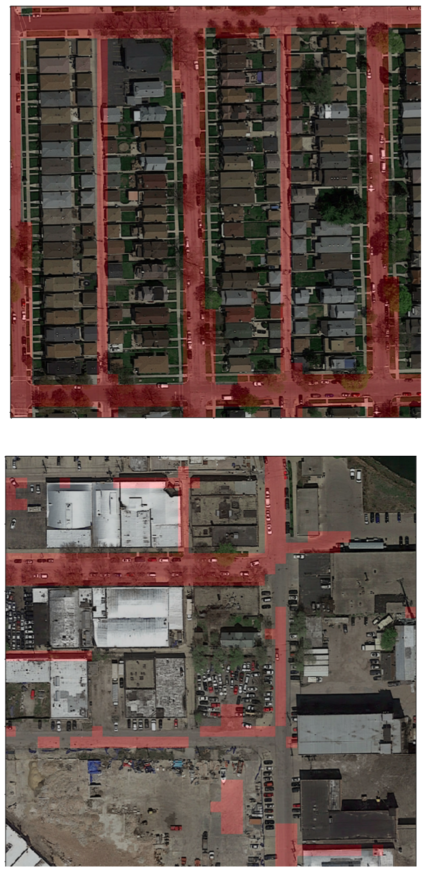

Fig. 6. Two predictions from the best model.

It is clear that the patches are very well classified in the first one and the predictions are close to what a real human would categorize. However the second one shows that there are still some imperfections that can be improved, especially for the roads which are neither horizontal nor vertical.

With regard to a possible future work to improve the obtained results, we did not explore any alternative to the threshold of 0.25 when assigning the labels to the patches because it seemed to be imposed by the examples provided in the subject. However that threshold plays a crucial role in the labels distribution and could be considered as a hyperparameter. Another way to get more training data could also be to play with the stride parameter which remained 16 in the whole project because the tiles' size was 16x16. But reducing the stride gives more data without any transformation. We are convinced that the U Net should also give better results because that kind of neural network is very well adapted to image segmentation problems.

References