Machine Learning: Road Segmentation Report

Vincent Petri
Mikael Morales Gonzalez
Charles Parzy–Turlat

***Abstract*—This is the report for the road segmentation project of the Machine Learning course, taught to master students during the Fall semester 2017.**

I. Introduction

The goal of road segmentation is to be able to segment, or recognize roads in a given scene. One can find lots of applications in computer vision. It is used in autonomous driving vehicles or pedestrian detection system for example. The aim of the project was to build a model capable of segmenting roads from the rest of the image. The training set provided is composed of one hundred pictures, 400x400 pixels each, with their grountruth. A grountruth is basically a picture which is white where there is road on the original picture, and black anywhere else. This report summarizes our work on the different methods that can be used in order to achieve road segmentation. First of all, we built a model using support vector machine (SVM), using sklearn. Then, we created a convolutional neural network, using Tensorflow and Keras. These two methods will be presented in the next sectioncovered.

II. Models and Methods

Our primary goal for this project was to build a convolutional neural network, but in order to have a comparison model, we decided first to implement a non-linear classifier using support vector machine (SVM) and the kernel trick. Afterwards, we implemented a convolutional neural network (CNN), and used the results obtained with SVM to compare the robustness and the accuracy of the CNN. However, it is worth to notice that comparing the two models is not fair, since much more work was involved during the implementation of the CNN, so SVM should be seen as a baseline model.

A. Support Vector Machine

1) Features: Each image was decomposed into 16 * 16 pixels patches. The first features we used for each patch was the mean and the variance for the different RGB channels. That represents a total of 6 features. We then tried to add some context for a patch by adding the features of its eight neighbours. We got better results using only the means of the neighbours' channels so we only kept these extra features. That represents a total of 30 features per patch.

2) Parameters:

Kernel Function: Several kernel functions were used to test the accuracy of the model. To find the most accurate one, we did a grid search with three different kernel functions: radial basis function, polynomial, and linear. The most accurate one turn out to be the radial basis function, the result seems logic given the dataset, as you can see below.

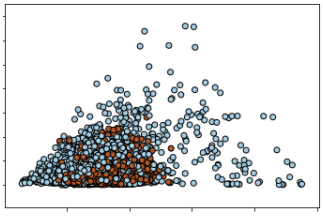

Figure 1: 2D representation of 10 images features with their corresponding classification.

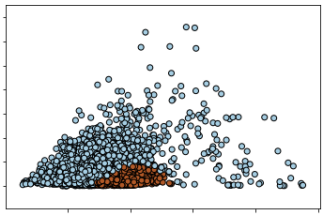

Figure 2: 2D representation of 10 images features predicted using SVM with RBF kernel.

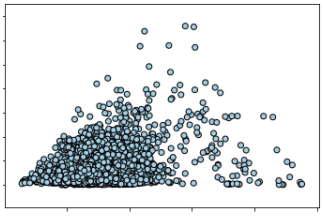

Figure 3: 2D representation of 10 images features predicted using SVM with a polynomial kernel of degree 3.

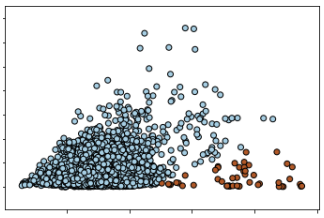

Figure 4: 2D representation of 10 images features predicted using SVM with a linear kernel.

Penalty parameter C: The same method was used to find the best penalty parameter for SVM, we did a grid search on the range [1e-5, 1e7] and the best value found was 1e6. The value found is relatively large, which means that the model will choose a smaller-margin separating the hyperplane, such that most samples are classified correctly. Otherwise a small value of C would have caused the model to look for a larger-margin separating the hyperplane, even if it would have misclassified some samples.

3) Results:

Accuracy: Using this method we managed to reach a score of 0.78789 on the kaggle competition. We can see on Figure 5 that our classification is not really robust nor accurate as it sometimes missclassify roads as background and some parking lots or roofs are classified as roads.

Figure 5: Classification using SVM.

Limit: To reach a good accuracy with SVM, we had to apply lots of preprocessing on the pictures before training our model on them. So we applied some convolutional matrices on the dataset of images, for instance a gaussian blur. The results we got were not convinced, we probably should have chained convolutional matrices in order to have a more accurate result. But it would have required too much work, knowing that the CNN would give us better results with much less manual preprocessing, so this model was kept as a baseline.

B. Convolutional Neural Network

1) Data Augmentation:

Window: Two different methods to generate inputs were tested. Each one was using 16x16 pixels patches, whose features are the RGB channels of all the contained pixels.

- No preprocessing: each patch was given as is, without any data augmentation.
- Window: each patch was increased with every neighbours patches at distance smaller or equals to 2. The window in composed of 25 patches, whose label corresponds to the center's one. This technique allowed us to take the context surrounding a patch into account.

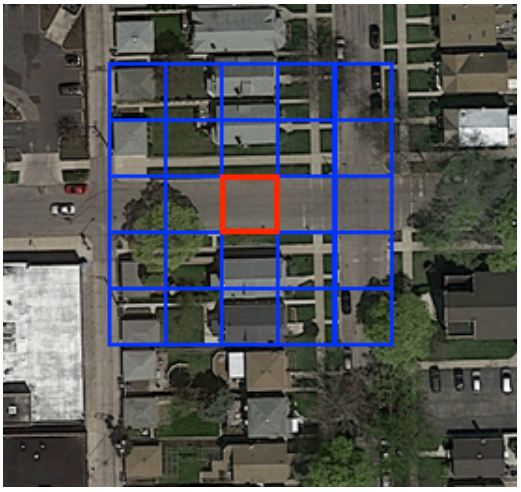

Figure 6: Window representation with its 25 patches.

Mirror Boundaries: To be able to apply the window described before at the boundaries of the pictures, we need to add some padding at the boundaries. A possible solution could have been to pad the images with black pixels, but we decided to take advantage of the structure of a road, which is generally continuous. So, we augmented each picture by a reflection of its boundaries.

Figure 7: Image with mirror padding.

Rotation of Image: To improve the robustness of our model, we trained it on a larger number of labelled data. To increase our training dataset, we added the 90, 180 and 270 degrees rotations for each image. The training dataset that contained 100 images initially, contained 400 images after adding the rotations.

2) Model:

Activation Function: We used the ReLu activation function at all layers except the last one, for which we used softmax function. ReLu is one of the most widely used activation function in CNN, because it prevents its gradient from vanishing which is a nice property since the training is done by stochastic gradient descent. And it is also efficiently computed. Softmax outputs the equivalent of the probability of being a road or not.

Model Used: The structure used is the most found one in the literature about convolutional neural network [1]. It consists of several groups containing a convolutional layer with a pooling layer, followed by fully connected layers. Convolutional layers applies convolution filters to the patches. The pooling layer reduces the number of samples to increase the computation speed. The fully connected layer classifies the extracted features from previous layers. To avoid overfitting, we added dropout layers after every pooling layers and the first fully connected layer. Parameters such as kernel size or strides were empirically found. We used categorical cross entropy as the loss function since the literature recommends this loss function with the softmax output activation function, since it is less time consuming computational-wise. We used the Adam optimization algorithm [2] which is a more efficient extension to stochastic gradient descent. The number of hidden layers was empirically tested. A summary of the structure is showed in Table I.

Layer	**Description**
Convolutional Layer	Input shape: 80x80x3 Output Dimension: 128 Kernel: 5x5 Strides: 2x2 Activation Function: ReLu
Max Pooling Layer	Pool Size: 2x2
Dropout Layer	Rate: 0.25
Convolutional Layer	Input shape: 80x80x3 Output Dimension: 256 Kernel: 3x3 Strides: 1x1 Activation Function: ReLu
Max Pooling Layer	Pool Size: 2x2
Dropout Layer	Rate: 0.25
Flatten Layer	Flattens the input
Fully Connected Layer	Output Dimension: 256 Activation Function: ReLu
Dropout Layer	Rate: 0.5
Fully Connected Layer	Output Dimension: 2 Activation Function: Softmax

Table I: Structure of the Model

Training Validation: To verify the accuracy of our model, 20% of the dataset has been dedicated to a validation set. We decided not to do a k-fold cross validation because it would have increase the training time too much.

3) Post Processing: There were some isolated patches labelled as a road in the middle of patches labelled as non road, and reciprocally, some patches labelled as non road in the middle of patches labelled as road. The post processing consists in finding these exceptional patches and label them like their neighbours.

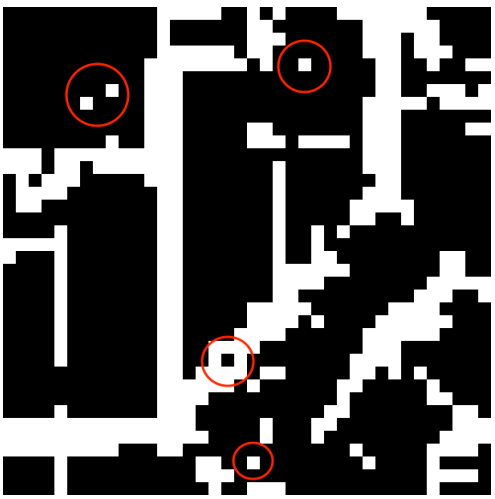

Figure 8: Classification of the CNN model without post processing.

Figure 9: Classification of the CNN model with post processing..

4) Results: These results shows the different step we took to our final result. Each steps produced an improvement. Moreover, to obtain these results, we run our model on an Amazon virtual machine with the following specifications:

- 1 GPU: NVIDIA Tesla K80 12GB
- 61 GB of RAM
- 4 CPU

Methods	Accuracy
SVM	0.78689
CNN without window	0.87390
CNN with window (48x48x3) 4 rotations	0.89842
CNN with window (80x80x3) 4 rotations	0.90659
CNN with window (80x80x3) 4 rotations and post processing	0.90778

Table II: Accuracy of the different methods used

III. Conclusion

The prediction produced by our model are fairly accurate. We can also see that the model is robust. Indeed, as you can see on Figure 10 and Figure 11 that the trees don't hinder the recognition of the road. However, when road are neither horizontal nor vertical, we can notice the downside of using patches compared to a per pixel classification. An improvement of our model would be to use a per pixel classification. Instead of having a window per patch, we would have a window per pixel. But it would have generated 64 Millions samples, which is too costly computationally.

Figure 10: Classification using CNN model with post processing.

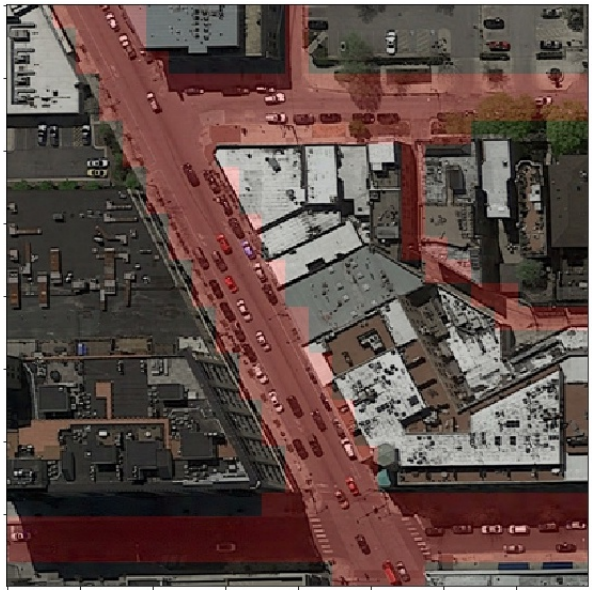

Figure 11: Classification using CNN model with post processing.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12, 2012, pp. 1097–1105.

[2] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980