Computational Intelligence Lab Project: Road Segmentation

Igor Pesic
Department of Computer Science,
ETH Zurich

Felipe Sulser
Department of Computer Science,
ETH Zurich

Minesh Patel
Department of Computer Science,
ETH Zurich

***Abstract*—Image segmentation of aerial road images has gained significant research interest in recent years due to numerous applications ranging from civil infrastructure to disaster preparation. In this work, we focus specifically on road segmentation, which is a simpler, but still very useful, subset of the general problem. Our goal is to accurately classify image pixels as part of either a road or the background. To this end, we have combined techniques proposed in several academic research papers and have tailored them to suit our requirements and resources as best as possible. Our solution uses a convolution neural network (CNN) in addition to techniques for data augmentation, feature selection, and post-processing. We find that our results are close to those of state-of-the-art solutions in current academic work despite our model being much simpler.**

I. Introduction

Image segmentation covers a general class of problems that require classifying input images' pixels into different regions, called *segments*, based on predefined criteria. *Road segmentation* is a subset of image segmentation that focuses on identifying all image pixels belonging to roads and has gained academic interest due to numerous real-world applications ranging from automated monitoring of civil infrastructure [1] to disaster preparedness [2].

Our goal in this work is to perform accurate road segmentation on aerial satellite images. Given a set of input aerial images, we would like to accurately determine which pixels belong to roads and which do not, assigning each pixel a value of 0 (no road) or 1 (road). Ideally, we would segment the images at a pixel granularity, but in order to simplify the problem, we use an approach that achieves patch-wise granularity. We find that our approach simplifies the problem substantially and is sufficient for this course project.

Our solution makes use of a convolutional neural network (CNN) combined with pre- and post-processing steps in order to achieve high-accuracy road segmentation. To the best of our knowledge, we have designed a novel architecture, which combines multiple academic works on similar topics [3]–[6] and extends the combination by adding our own ideas. Although we focused primarily on the design of our CNN, we also invested a substantial amount of effort into post-processing the outputs using a denoising step in order to improve the final results.

This report has the following structure: in Section II we explain the preprocessing we do before training and evaluating the model. In Section III we describe our model, including both the CNN and the denoising applied afterwards. Finally, in IV we discuss the results we achieve using our model.

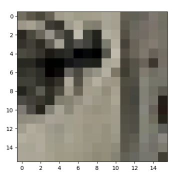
II. Data

A. Data Augmentation

The original data set consisted of 100 labeled aerial images of road maps. As our approach consists of a deep neural network, we consider that the given data is not enough and thus we try to augment it to expand it further. For data augmentation, we first rotate all the images by 90 degrees and to mirror them. Beside the obvious increase in the training data size, this approach also makes our model more robust. Furthermore we have noticed that the predictions on the diagonal roads were much worse than on the horizontal and vertical ones. This appears to be due to the lack of diagonal roads in our original data set. In order to fix this, we pick 9 images with diagonal roads and highways which we rotate again by 180 and 270 degrees. Finally, we have noticed some inconsistencies in the labeling of some images in the dataset and we have decided to discard those from our training data (i.e. a building classified as a road and viceversa). At the end, our data set has 309 images in total, created by rotations and by augmentation of diagonal roads.

B. Feature extraction

In the baseline model we split the image in patches of size 16×16 pixels. This provided granularity that was good enough, but it lacked the information about the surroundings. In order to address that, we have come up with solution that we called *added context*. We have also found that other authors propose a similar approach to enhance the context of a patch such as in Mnih et al. 2010 [4] and Alina Elena 2016 [3]. The approach, that enhances the context of a patch, first splist the image in 16×16 patches and then adds the surroundings area to it so that patches have the total size of 64×64 pixels. The difference between patches is shown in 1. Labels are based solely on the original 16×16 patch, however in order to classify them correctly, the input is augmented with the context of the small patch.


(a) Original patch

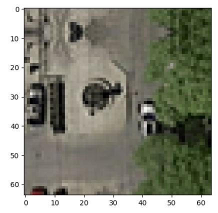

(b) Context added patch

Figure 1: Context augmentation for patches

We have chosen the patch size empirically by trying out different context-added sizes and 64 turned out to bring the best score on the Kaggle competition[1].

C. Class balancing

Because the dataset was very unbalanced (i.e. the number of background patches was approximately three times bigger than the number of road patches, we had to balance the data so that both classes have a similar number of datapoints. Balancing the data helps reduce the bias towards a certain value in the form of a prior, which is due to the error minimization we are performing. We have done that by equalizing the number of patches in both classes, i.e. by randomly picking a limited number of background patches.

III. Models and Methods

A. Baseline Model

The baseline model works on batches of 16×16 patches. Its configuration is as follows: $IN(3, 16 \times 16) - C(32, 5 \times 5/1) - MP(2/2) - C(64, 5 \times 5/1) - MP(2/2) - FC(512) - FC(2)$
Where:

- $IN(a, b \times c)$ – Input image of a channels and size $b \times c$
- $CONV(a, b \times c/d)$ – Convolution layer with depth a, window size $b \times c$ and stride d
- $FC(a)$ – Fully Connected layer of size a
- $MP(a/b)$ – Max Pooling layer of size a with stride b

The weights of the network are L2 regularized and the optimizer used is the Momentum Optimizer with an exponential decaying learning rate with a decay rate of 0.95 and starting at 0.01. Each convolutional layer is followed by a rectified linear unit ($RELU$) and the activation function in the Fully Connected layers are the identity function . In the end, the softmax function is applied to the output of the last Fully Connected layer.

B. Improved CNN Architecture

The core of our work is the CNN design. We have got the main idea for the architecture of the network from Alina Elena [3]. In this design, they connect both a VGG network [5] and an AlexNet network [7] to create a dual stream network that takes as input the local patch to be predicted and the whole image as context. Our solution takes this network as inspiration, however it only takes the local information as input. This decision was done due to limited data and also because empirically the results obtained with it were good.

Our CNN network can be described as follows: $IN(3, 64 \times 64) - C(64, 3 \times 3/2) - MP(2/2) - C(128, 3 \times 3/2) - MP(2/2) - C(256, 3 \times 3/2) - MP(2/2) - C(512, 3 \times 3/2) - MP(2/2) - FC(2048) - FC(2048) - FC(2)$

Additionally all convolutional layers are followed by rectified linear units ($RELU$) layer. The output of the CNN is the softmax function for the two classes and we use the Adam optimizer [8].

To reduce overfitting, we initialize the parameters of the network using xavier's initialization algorithm [6] and we also use a dropout rate of 0.5 on all the Fully Connected layers during training.

C. Error function

Our CNN model reduces the log-loss, but for the evaluation we have used two other loss metrics. The first one was the classification error and it was used on the validation set that helped us find the right number of training epochs. Further discussion on that will follow in the next section. The second one was the F-1 score to calculate the test error. The latter was given by the Kaggle competition.

D. Model hyper-parameters

Since we used the Adam optimizer, there were not many parameters to tune. One parameter was the batch size, which we have not changed from the baseline model since in the one of previous exercises we have learned the it should nether be too big nor too small, so we found 32 to be a good choice. The only other parameter to tune was the number of training epochs. This one we have empirically chosen to be 300 based on training the model on 95% of the images and validating it in every epoch on the remaining 5% of the images. The Figure 2 shows how the validation error changes with the number of epochs. Beside these, there were no further hyper-parameters to tune.

E. Post-processing

In order to improve the output obtained from the CNN network, we perform a post-processing step that helps reduce the noise of the output image and correct some prediction errors. Errors may appear in the image due to the inconsistencies in the roads. As an example, objects that overlap with roads such as cars or trees might have a negative effect on the classification due to their difference in color and shape in contrast with the road. Furthermore, structures like rooftops might sometimes appear like roads.

[1]inclass.kaggle.com/c/cil-road-segmentation-2017

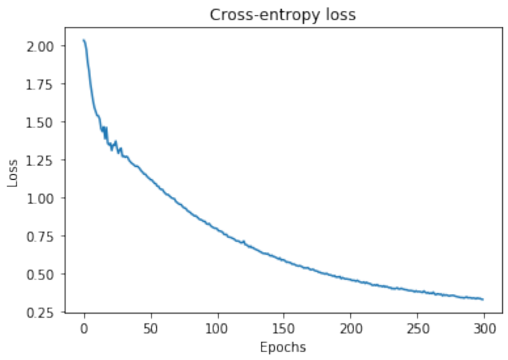


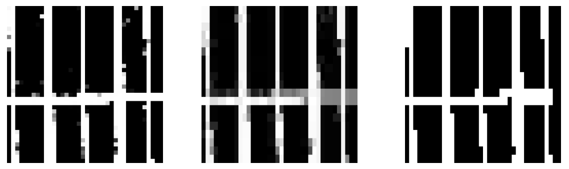
Figure 2: Cross entropy loss with 5% validation set

1) Model Description: We propose an approach that first denoises the image and then performs a prediction on the patches. Given the output of the CNN as a grayscale image, we perform a denoising using wavelets. Following the denoising, we convert the image to binary representation (either black or white patches) and then we perform a prediction of the *border patches* using a Multilayer Perceptron (MLP). We tried other models such as SVM (or SVC for classification) and Random Forest, however we obtained the best results with the MLP. Finally, we change the patch's color if at least 7 out of 8 neighbors have a different color, because we believe that it is highly likely that it is a misclassified patch.

In figure 3 we can see the outputs of each step. We start with the raw CNN output, we then apply the wavelet denoising, and afterwards we apply the MLP to the border patches only and fill the patch color with the neighbouring technique.

2) Wavelet denoising: For the denoising, we choose wavelets because the results obtained were smooth and appropiate for road images. The wavelets used are Daubechies 1 wavelets. Furthermore the denoising is done assigning the hyperparameter sigma to 3.

3) Multilayer Perceptron classifier: The input datapoints of the classifier are patches surrounded with context. For the patches, we consider a window of 11×11 patches, with the current one to be classified centered. Also, we do not classify all the patches in the image, only the *border patches*. A patch is considered a *border patch* if 5 of its 8 immediate neighbors have a distinct color. We only consider these patches because we observed that the other patches are generally correctly classified as the raw output of the CNN is not so noisy compared to the baseline's output.
For the structure of the classifier itself, the parameters were found using a cross validated grid search. The best settings found are to use two hidden layers of size 50 each and to use the logistic activation function. Finally the training of the MLP was done on the 309 groundtruth images that we obtain using data augmentation techniques.



(a) Raw CNN output (b) Wavelet denoising (c) Wavelet + MLP + Neighbor

Figure 3: Post-Processing steps

IV. Results

A. Implementation

The model presented here is implemented using tensorflow for the CNN and the denoising was developed using scikit-learn. The CNN was trained on an Microsoft Azure NV6 instance for 20 hours and 52 minutes using one kernel of the Nvidia Tesla m60 GPU using 2048 CUDA cores provided by the instance. Furthermore the post-processing part was also trained on the Azure instance. The training of the MLP using 6 jobs was done in 2 hours and 51 minutes for the Grid Search method in order to find the optimal parameters.

B. Score

C. Comparison to baseline model

The baseline model achieves an F1 score of 0.73723 on the public Kaggle competition, which is significantly lower than the one obtained by our model. Furthemore, the output images obtained by the baseline model are noisy, with patches of roads completely isolated which is something that does not happen on roads.

V. Conclusions

In this paper we have discussed a novel method to solve road segmentation on satellite images. The relatively deep CNN presented seems to be a good compromise between ease of train for small dataset and accuracy. Furthermore, post-processing is an important step for road segmentation. The prior knowledge of the input images (roads), allows the training of models for image denoising that take into account the overall shape of the roads.
Regarding the score, despite the lack of input data it achieves in general high score close to state-of-the-art methods. However this could be improved by making the network more deep and with a further increase in the dataset, something which is left as an idea for further work.

References

[1] S. C. Radopoulou and I. Brilakis, "Improving road asset condition monitoring," *Transportation Research Procedia*, 2016.

[2] Y. Li, J. Li, and M. Chapman, "A fuzzy relational method for image-based road extraction for traffic emergency services," *Geomatics Solutions for Disaster Management*, 2007.

[3] A. E. Marcu, “A local - global approach to semantic segmentation in aerial images,” 2016, master Thesis.

[4] V. Mnih and G. E. Hinton, *Learning to Detect Roads in High-Resolution Aerial Images*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 210–223. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15567-3_16

[5] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[6] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[8] 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