Road Extraction from Aerial Images

Delio Vicini, Matej Hamas, Taivo Pungas
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—In this paper, we present a novel method to automatically segment aerial images into road and non-road patches. Our technique classifies small image patches using a deep convolutional neural network. On top of the neural network, we apply a post-processing filter based on a support vector machine classifier. The combination of a convolutional neural network and a post-processing filter yields a prediction accuracy of over 90% on a representative benchmark data set.**

I. Introduction

Automatically extracting roads from aerial images is a long-standing research topic in computer vision. The solution to this problem has potentially many applications ranging from automation of map making to urban planning and environmental monitoring.

Given an RGB aerial or satellite image, we seek to classify each 16×16-pixel patch as *road* or *non-road*. In the dataset we used, a distance of 1 pixel in the image corresponds to roughly 0.3-0.4 meters.[1] This problem is an instance of a binary classification problem. The training data we use to train a binary classifier contains aerial images with corresponding ground truth per-pixel road masks, as shown in Figure 1.

As aerial image segmentation is a long-standing problem in computer vision with many applications, there is a rich body of previous work in this area [1], [2], [3], [4], [5]. Previous approaches include support vector machines (SVMs) on manually extracted features [1], convolutional neural networks (CNNs) [3], [5], conditional random fields [4] and other probabilistic approaches. Most of the past work has concentrated on per-pixel segmentation. Some papers also look at multi-class semantic classification, e.g. introducing a specific *building* class [5].

Unlike most of previous work, we concentrate on per-patch and not on per-pixel prediction. We have developed a novel solution by combining a deep CNN with an SVM-based post-processing algorithm.

We compare our approach to two different baseline algorithms. One of them is a 2-layer CNN with stochastic gradient descent (SGD) optimizer. The second is the same CNN followed by an dictionary based denoising [6] step.

Our main contributions are a patch-based, 4-layer CNN architecture for road segmentation and a SVM-based post-processing scheme which further refines predictions produced by the CNN.

[1] Since the used dataset does not come with any scale information, we infer the scale roughly from the size of the observed cars.

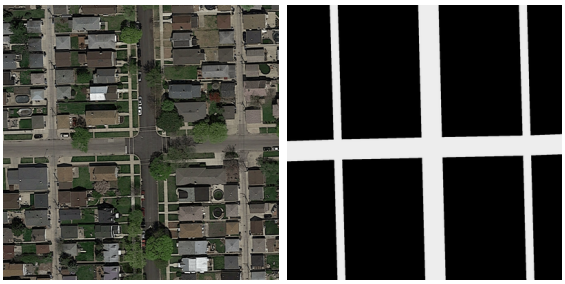


Fig. 1: Aerial image of an urban scene and the ground truth binary classification displaying the roads.

II. Models and Methods

In this section we describe data augmentation techniques, baseline and final CNN architectures and post-processing techniques applied on the top of the CNN output.

A. Data Augmentation

The lack of training data is a universal problem in machine learning. We extract patches of a particular size from input images. To enlarge the dataset, we generate more training patches by rotating the image 90 degrees counter-clockwise.

Subsequently, we zero-mean the training dataset by subtracting the mean image from all patches. We do not divide by standard deviation as the images are naturally well distributed.

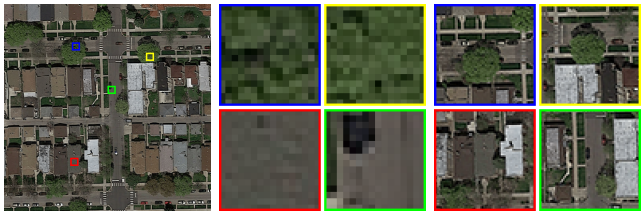
We then generate the corresponding ground truth patch labels from the given ground truth images. Each ground truth patch is converted to the label 1 if more than one quarter of the pixels in the patch are road pixels; otherwise the patch label is 0. The labels are stored in a 1-hot format, i.e. as tuples [0, 1] or [1, 0].

Finally, we balance the training dataset to include the same number of positive and negative examples. This is done to prevent the CNN from learning a bias towards one label.

B. Notation - CNN Architecture

Here, we provide a short overview of CNN layer types that are used in the following sections.

- $\mathcal{C}$(N×N, I, F) - Convolutional layer with filters of size N×N, I input channels and F different filters. The number of different filters corresponds to the number of output channels. The default stride is 1 both vertically and horizontally and a clamping boundary condition is used.
- $\mathcal{M}$(N×N, S) - Max-pooling layer of size N×N, stride S.
- $\mathcal{LRN}$ - Local response normalization layer across the current batch of training data.



(a) Input image (b) 16×16 patches (c) 64×64 patches (extracted from the downsampled image)

Fig. 2: Aerial image of an urban scene and some example patches. Note that in the 16×16 patches it can be impossible to infer the correct labeling due to the limited perception of the surroundings.

- $\mathcal{FC}$(I, O) - Fully connected layer from I input channels to O output channels.

C. Baseline CNN Architecture

The baseline CNN operates on patches of size 16×16. It has two convolutional and two fully connected layers with the following architecture:

$$\mathcal{C}(5\times5,\ 3,\ 32)\ -\ \mathcal{C}(5\times5,\ 32,\ 64)\ -$$
$$\mathcal{FC}(1024,\ 512)\ -\ \text{ReLU}\ -\ \mathcal{FC}(512,\ 2).$$

Each convolutional layer is followed by a rectified linear unit (ReLU) activation layer and a $\mathcal{M}(2\times2,\ 2)$ max-pooling layer.

The activation functions in the FC layers are identities. We apply *softmax* to two outputs to get the probabilities for both classes. All the weights are initialized normally with standard deviation 0.1. The weights in the FC layers are L2 regularized with a factor 5×10^{-4}. We use a plain vanilla stochastic gradient descent (SGD) optimizer with an exponentially decaying learning rate, starting at 0.01 with a decay rate 0.95. Training samples are processed in batches of 16 samples.

D. Improved CNN Architecture

We modified the baseline CNN in various ways to improve its performance. We made it deeper, experimented with the widths of layers and filter sizes, tried using dropout or local response normalization instead of L2 regularization of FC weights, tested different optimizers (momentum and Adam [7]) and learning rates. We also implemented cross-validation to estimate the training time after which the out-of-sample error converges and subsequently trained the CNN for at least that amount of time.

Moreover, we increased the context size to 64×64, i.e. the network predicts a label for the central 16×16 patch using more context. Image boundaries are handled using mirror boundary conditions. Additionally, we downsample training and testing images by a factor of two. This means that the context considered by the neural network effectively doubles. There is no need to use the full high resolution of the input images to reliably predict 16×16 patches.

Figure 2 shows a comparison between the context size of the baseline CNN and the context size of the improved CNN. The yellow and blue patches both show a part of a tree and are extremely similar. However, the blue patch belongs to the part of a tree which covers a road and should be labeled as road, while the yellow patch is not covering the road. The correct labeling can only be inferred by considering a larger context. The red patch could locally easily be mistaken for a road, as its color is very similar to the color of most streets. Even the green patch is hard to predict from the small context, since cars can also occur in non-road areas (e.g. parking lots).

The final architecture of the improved CNN uses four convolutional layers and is structured as follows:

$$\mathcal{C}(5\times5,\ 3,\ 16)\ -\ \mathcal{C}(3\times3,\ 16,\ 32)\ -$$
$$\mathcal{C}(3\times3,\ 32,\ 32)\ -\ \mathcal{C}(3\times3,\ 32,\ 64)\ -$$
$$\mathcal{FC}(1024,\ 64)\ -\ \text{ReLU}\ -\ \mathcal{FC}(64,\ 2).$$

After every convolutional layer, we apply ReLU, $\mathcal{LRN}$ and $\mathcal{M}(2\times2,\ 2)$ max-pooling.

The activation function in the first FC layer is the identity function and in the second the sigmoid function. We use neither dropout nor L2 regularization of the FC weights as the local response normalization already sufficiently prevents overfitting. The class probabilities are again obtained using *softmax*. Cross entropy is used as the loss function.

After discovering that exponentially decaying learning rate does not help, we fixed it to the empirically determined value 0.01. In the final model, we use the Adam optimizer algorithm [7] with $\epsilon = 0.1$ and the batch size is 32. All the weights are initialized normally using standard deviation 0.1.

E. Post-Processing

The convolutional neural networks output independent predictions for all 16×16 patches of the processed image. The spatial arrangement of predictions however contains valuable information, which can be used to further improve the prediction accuracy. For example, it is highly unlikely to observe a road which only covers one patch. If the CNN predicts an isolated patch as belonging to the road label, we can discard this prediction with high certainty. The opposite also holds: a patch labeled as non-road surrounded by patches labeled as road is most likely part of the road as well.

The simplest post-processing scheme is therefore to simply reject outliers based on the predictions in a 4-neighborhood around the current patch. If all four neighbors of a patch are assigned the label road, we can also assign the label road to the current patch. A simple scheme like this can already significantly improve prediction accuracy, because many prediction errors are isolated outliers. However, more sophisticated algorithms promise even more gain in quality, as the post-processing algorithm should account for more structure than simply the four closest neighbors.

One more advanced way of reducing the noise in the CNN output is to use dictionary based denoising [6]. We do this by learning a dictionary of 100 atoms, each representing a 5×5 square of patch predictions. The dictionary is learned

from the patch labels in the training data using the algorithm presented by Mairal et al. [8]. Given the trained dictionary, we can then denoise the predictions for a test image by solving an orthogonal matching pursuit problem for each 5×5 patch of patch predictions. This is done in a sliding window fashion, similar to the work by Elad et al. [6]. We found this to work quite well if the level of noise in the CNN predictions is high, which is the case in the baseline CNN. However, as the quality of the CNN improves, dictionary based denoising becomes less useful.

In our pipeline, we filter the CNN output by predicting the label of a patch from the labels of the surrounding patches using a binary support vector machine (SVM) classifier. Given a window of 7×7 patches predicted by the CNN, we predict the label of the central patch using a SVM classifier. We directly use the per-patch probabilities produced by the CNN as features for the SVM classifier. We also tried to automatically extract features using Restricted Boltzmann Machines [9], but could not achieve an increase in prediction accuracy.

The SVM uses a radial basis function kernel and a soft-margin penalization weight of 1. We did not perform a systematic search for optimal SVM hyperparameters, as the used parameters seem to work reasonably well.

We found it beneficial to iteratively apply this denoising technique. This seems to help to propagate confident predictions, as suggested by Mnih and Geoffrey [10]. In practice, we use only two iterations. After two iterations most noise is already removed. Increasing the number of iterations biases the predictions too much and does not lead to a gain in accuracy.

We also tried using graph cut based inference [11]. For this we computed per-pixel labels by accumulating CNN per-patch votes using a sliding window. Graph cut based image segmentation however did not work well, since it relies on strong edges between fore- and background. In our aerial images, this is not the case and the boundary between road and non-road is fairly weak in terms of local edge contrast. Furthermore, the probabilities produced by the CNN are often close to 0 or 1. Graph cut based segmentation is simply not robust enough towards a locally strongly biased data term.

We also experimented using an additional CNN for post-processing, as suggested by Mnih and Geoffrey [10]. We used a network with two convolutional and two fully connected layers, predicting the central patch from a 9×9 square of patch predictions from the first CNN. However, we did not get an improvement in accuracy over the simple SVM based post-processing scheme.

III. Results

The neural networks were implemented using TensorFlow [12] and the post processing using scikit-learn [13]. The training set consists of 100 images of a resolution of 400×400 pixels. For the baseline CNN, we used the code provided in the lecture materials, without any modifications. We trained it for around 30 minutes, which was sufficient to achieve convergence. The training was done on a cluster node with 16 cores (Intel Xeon E5-2697 v2 or Xeon E5-2680 v3 depending on availability on the Euler cluster) and 128GB of memory. We train our improved CNN model for around 20 hours on the same hardware, which is sufficient to achieve convergence of the loss function. The post processing SVM trains for around 1.5 hours on a conventional desktop computer (Intel Core2 Quad q9550, 4GB of memory). All neural network code runs exclusively on the CPU, since the used cluster does not provide any GPU nodes. Given the trained models, individual satellite images are processed in a matter of seconds.

We compare our method to the two different baseline implementations described previously. The first is the CNN described in Section II-C. The second is the same CNN combined with dictionary-based denoising. We measure prediction accuracy as the reported F_1 score in the public Kaggle leaderboard. Ideally, one would compute an average cross validation error, to also have an estimate of the variance of the measured accuracy. However, since CNNs are very expensive to train we decided to report the Kaggle score only.

The baseline CNN achieves an F1 score of only 0.7765. Note that assigning the non-road label to all patches already gives a socre of 0.7303. Post-processing the output from the baseline CNN using learned dictionaries improves the F1 score to 0.8256.

Our improved CNN on its own achieves an F1 score of 0.8710. Applying our SVM based post-processing procedure on top of these outputs, we reach a score of 0.9020. Note that applying the dictionary based denoising from the baseline we get an accuracy of only 0.8758.

IV. Discussion

The baseline CNN clearly suffers from the fact that its context size is too small. Given a 16×16 image patch at the resolutions we worked with, it is oftentimes impossible, even for humans, to correctly decide whether this patch belongs to a road or not. The results of the baseline CNN are therefore very noisy, as seen in Figure **??**.

Adding dictionary based denoising on top of this baseline CNN improves the predictions. The dictionary based method manages well with isolated outliers, as shown in Figure **??**. However, the prediction quality is inherently limited by the output of the underlying CNN. In contrast to image denoising, the noise produced by the CNN does not have zero-mean. Oftentimes the CNN will mispredict a whole group of patches. The dictionary based denoising then fails to infer the correct labels.

The accuracy gain we achieve using our improved CNN is mostly due to the increased patch size. The larger context reveals much more information about the currently processed image patch. The result shown in Figure **??** shows much less noise than the output from the baseline CNN. Despite the improved architecture, there are still many mispredictions.

Our post-processing algorithm manages to further reduce error. The result in Figure **??** shows that almost all noise has been removed. There are still some cases where the post-processing is not very successful. It has a strong bias for roads aligned with the image coordinate axis, since the test

(a) Raw CNN predictions (b) Post-processed predictions

Fig. 4: Post-processing results for diagonal roads. **TODO: ADD PROPER PICTURES**

set is dominated by these road directions. We experimented with rotating the training images by $45°$, but did not succeed in improving the prediction accuracy. Our current algorithm often predicts diagonal roads too thin or even removes them partially; an example of this is shown in Figure 4. This is the most prominent limitation of our current approach.

V. Summary

In conclusion, we presented an algorithm to automatically extract roads from aerial images. Our algorithm significantly outperforms the two baseline algorithms against which we compared and achieves an F1 score of slightly over 0.9. It is particularly strong in images containing mostly roads aligned to the image axes.

Our algorithm performs worse with images containing mostly diagonal roads. We believe that adding more training images containing diagonal roads should help reduce this problem. However, the parameters of the neural network and the post-processing then most likely need to be adjusted.

Another possible direction of future work would be to jointly predict a set of patches using the CNN instead of independently predicting patches. This could make post-processing less important, since each patch prediction would be influenced by the predictions of the neighboring patches.

References

[1] C. Huang, L. S. Davis, and J. R. G. Townshend, "An assessment of support vector machines for land cover classification," vol. 23, no. 4, pp. 725–749, 2002. [Online]. Available: http://pubs.er.usgs.gov/publication/70024767

[2] V. Mnih, "Machine learning for aerial image labeling," Ph.D. dissertation, University of Toronto, 2013.

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014. [Online]. Available: http://arxiv.org/abs/1411.4038

[4] J. A. Montoya-Zegarra, L. Lubor *et al.*, "Semantic segmentation of aerial images in urban areas with class-specific higher-order cliques," Photogrammetric Image Analysis, Munich, 2015.

[5] S. Saito and Y. Aoki, "Building and road detection from large aerial imagery," pp. 94 050K–94 050K–12, 2015. [Online]. Available: http://dx.doi.org/10.1117/12.2083273

[6] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[7] 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

[8] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. ACM, 2009, pp. 689–696.

[9] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations*, D. E. Rumelhart. and J. L. McClelland, Eds. MIT Press, 1986, ch. 2, pp. 194–281.

[10] V. Mnih and G. E. Hinton, "Learning to detect roads in high-resolution aerial images," in *Proceedings of the 11th European Conference on Computer Vision: Part VI*, ser. ECCV'10, 2010, pp. 210–223.

[11] Y. Y. Boykov and M. P. Jolly, "Interactive graph cuts for optimal boundary amp; region segmentation of objects in n-d images," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 1, 2001, pp. 105–112.

[12] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: http://tensorflow.org/

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Road Extraction from Aerial Images

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Vicini	Delio
Bungas	Taivo
Hamas	Matej

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 1.7.16	D. Vicini

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.