Road Extraction from Aerial Images

Krzysztof Lis, Isa Roesch
Group: Augurs
Computational Intelligence Lab 2016, ETH Zurich, Switzerland

***Abstract*—We present an implementation of a convolutional neural network trained to extract roads from aerial images. Our algorithm does not only focus on the area of the image to be classified, but also takes the surrounding image into account. To face the problem of a small given training set, we use data augmentation like rotation and noise to artificially enlarge the data our network can train on. The evaluation on a predefined test set shows that our algorithm performs considerably better than the baselines we compare it to and is even able to discover roads that are not marked by the ground truth.**

I. Introduction

Extracting roads from aerial or satellite images by hand is a tedious task. Automating this process using a convolutional neural network (CNN) is much faster, cheaper and can even be more accurate.

We have created a solution using a CNN to classify 16x16 pixel patches of given RGB images as road or non-road. The goal is illustrated by example input in Figure 1 and corresponding output of our algorithm in Figure 2.

It is not sufficient to consider only the content of the patch being classified because a grey roof of a building block could look exactly like a part of road. In our solution, the whole image is used as input and the CNN has a depth of 8 layers, so the output for a single patch is influenced by a significantly bigger area of the image and network has the opportunity to learn that regions containing roads are connected and often form straight lines.

Another challenge was the small set of available training data and in order to avoid overfitting we have extended the training set by employing data augmentation.

A. Task Description

As this solution was developed to participate in a competition, we give an exact formulation of the given task.
For this problem, 100 aerial RGB images of size 400x400 acquired from GoogleMaps were given as a training set. For each of these frames, a corresponding ground truth image was provided where each pixel is labelled as either road or background. Our goal was to train a neural network to classify each 16x16 patch of any input image as a road or background. To test the accuracy of the trained classifier, a test set of 50 608x608 RGB images was provided.
For the metric to evaluate the submissions for the competition, a F1 score was used. The F1 score is given by

$$F1 = 2\frac{pr}{p+r} \quad \text{where} \quad p = \frac{tp}{tp+fp}, \quad r = \frac{tp}{tp+fn}$$

where tp is the true positive, fp the false positive and fn the false negative rate.

Fig. 1. Example input image from the test set.

Fig. 2. Results of classification of image from Figure 1 performed by our algorithm, patches classified as roads are highlighted.

II. Methods

A. Network architecture

Our CNN takes RGB images input and outputs the estimated probability that a given pixel belongs to a road.

The architecture of the network used in our solution is depicted in Figure 3. We decided to only use convolutional layers with rectified linear units (ReLU) as activation functions. According to Volodymyr Mnih's thesis [4, p. 31], those layer types and activation functions perform the best when dealing with aerial images. Furthermore, since the network consists of only convolutional layers, it can process input images of any size - a useful property given the different sizes of training and test set images, as seen in section I-A.

The operation performed by each convolutional layer can be written in a simplified form as:

$$a_{nc} = \max(0, a_{n-1} \star k_{nc} + b_{nc})$$

$$a_0 = \text{input image},$$

where: a_{nc} - channel c of output of the n-th layer of the network,
a_{n-1} - output of the $(n-1)$-th layer of the network,
k_{nc}, b_{nc} - convolution kernel and bias of the n-th layer and channel c. The probability of being a road is calculated using a softmax operation on the 2 output channels of the last layer.

The strided convolutions used in some layers cause the output to be smaller than the input image. This reduces the memory consumption and allows a batch of 8 training images to be processed in one step on our hardware.

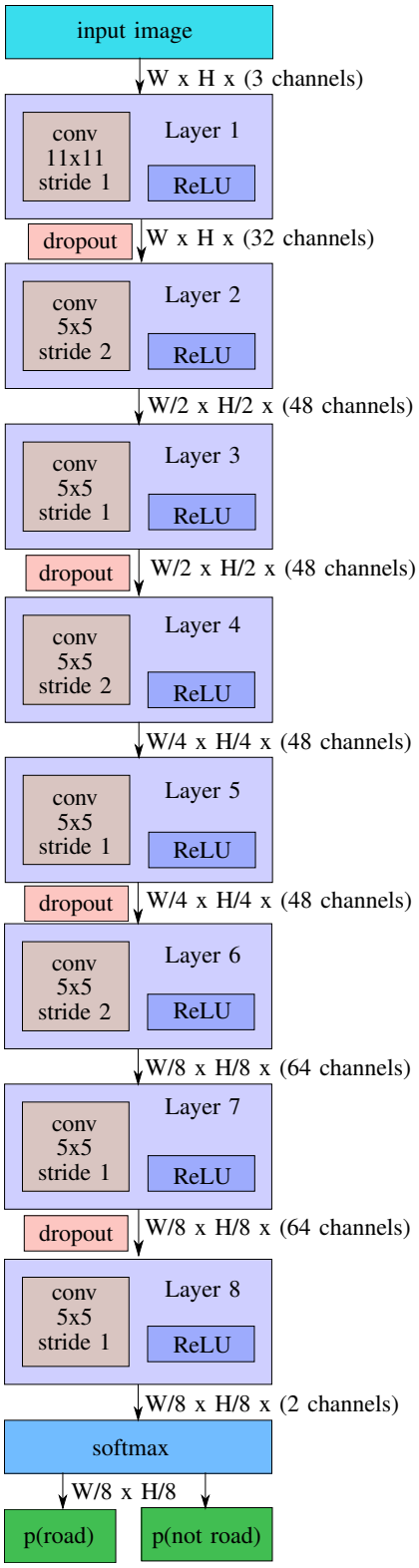

Fig. 3. Architecture of our CNN. Arrows show the direction of data flow. The output of a previous layer is the input for the next layer. Next to the arrows the dimensions of the data are written in format *width* x *height* x *channels*, with respect to the size of the input image - width W, height H. The convolution operation is described by the spatial dimensions of the kernels and stride - the number of pixels by which the kernel is moved over the input image to calculate the next output pixel.

Fig. 4. Sample aerial image

Fig. 5. Augmented version

B. Training

Training our CNN with the small given training set is a challenge as it easily leads to overfitting. To counter this effect, we use data augmentation to enlarge the training data. Additionally, we randomly drop out a percentage of neurons (set their output to 0) in certain layers during training. The latter is known as dropout [5] and was proven to be a valuable method to reduce overfitting.

We have divided the available training data into a training set (85%) and an evaluation set (15%) which is used to determine the stopping condition of the training.

1) Data Augmentation: To get more data out of the given training set, we have used the following methods for data augmentation:

- Mirroring the image on both axes
- Transposing the image
- Replacing a random amount of pixels by zero
- Rotating the image and rescaling it to fit
- Adding Gaussian noise

On each frame of the training set was an initial symmetry augmentation performed which consists of mirroring the image on both axes and transposing the image, resulting in four frames in total per given training frame. Additionally, frames are rotated by a random angle with a certain probability and Gaussian noise is added to each pixel's intensity. The final augmented frames are then used to train the network. An example of an augmented frame can be seen in Figure 5.

2) Optimization: The loss value for a single pixel is defined as the cross entropy between the ground truth and the probability of the pixel belonging to a road calculated by the network. The loss function used for training is the mean of all pixel loss values across the whole image.

The initial values of the convolution kernel weights were drawn from a normal distribution with standard deviation $\sigma = 0.1$ and the bias values were initially equal to -0.1.

We minimized the loss function using the Adam stochastic gradient-based optimization algorithm which computes individual adaptive learning rates for the parameters being optimized [2].

After training on each 25 batches of 8 images from the training set, we calculated the value of the loss function on the evaluation set. The subsequent values of the loss function were very noisy, so we smoothed them with a Gaussian blur.

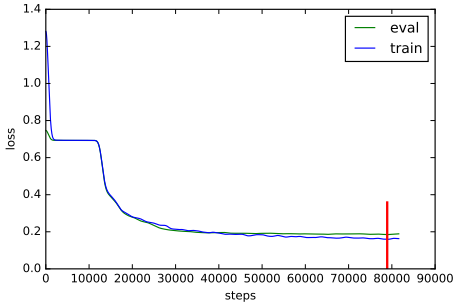

Fig. 6. Smoothed plot of the loss function on evaluation and training sets during training of the network. The red line shows the step with optimal loss values.

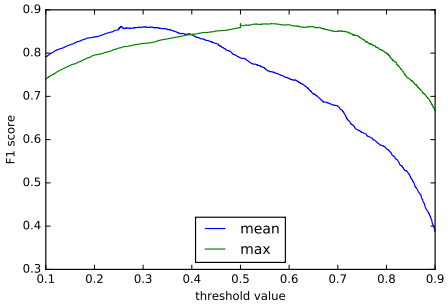

Fig. 7. The $F1$ score for different values of the discrimination threshold and feature, calculated on the evaluation set.

The training was stopped when the value of loss function had increased 2% above the minimum of all previously observed values. Then the trained weights corresponding to the best loss value were saved. The observed values of the loss function during training are shown in Figure 6.

C. Post-processing

For each 16x16 pixel patch to be classified, the network outputs 4 probability values which denote the probability of being a road for each 8x8 pixel patch. To perform the classification, we use a threshold of either maximum or mean of those probabilities across the patch. The algorithm chooses the threshold value and feature (max or mean) which yields the highest $F1$ score on the evaluation set.

According to the calculated scores - which can be seen in Figure 7 - the optimal threshold value is 0.5 and feature to threshold is max. The performance of the classifier is also illustrated using the ROC curve in Figure 8.

III. Results

The network was trained for approximately 80000 iterations on the training set which took about 20 hours on a Nvidia GeForce GTX 960 GPU.

To illustrate the features our network learned, we include an image of the convolutional kernel weights of the first layer 9. It is clearly seen that the network learned to recognize line features with different directions.

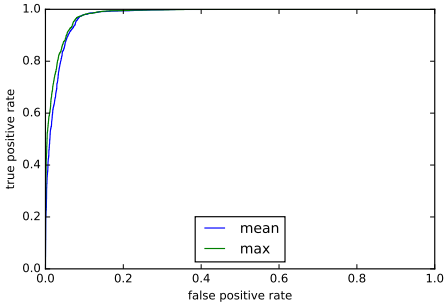

Fig. 8. Receiver operating characteristic plots of our classifier with respect to the discrimination threshold value for $mean$ and max of CNN output calculated over the classified patch. A bigger area under the curve suggests that better accuracy can be achieved by thresholding the max feature.

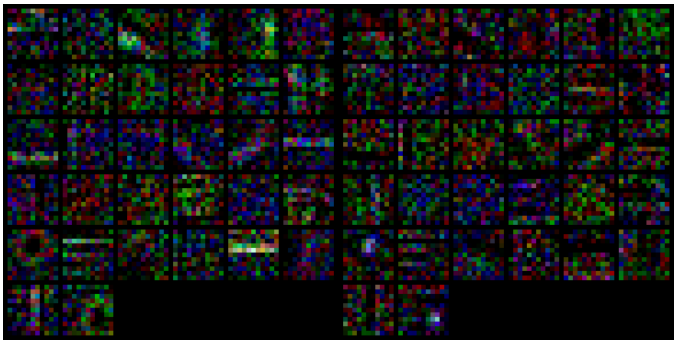

Fig. 9. Convolutional kernels of size 11x11x3 learned by the first layer of our CNN. The image is divided into positive weights (left) and negative weights (right).

A. Comaprison to baselines

We compare our algorithm to two existing baselines which are shortly described here.

CNN patch classifier A patch-based approach implemented as a CNN with two convolutional and two fully connected layers. This algorithm takes a 16x16 patch as input and outputs the calculated probability that the patch contains a road. It uses cross-entropy loss as the loss function and momentum optimizer [1] with an exponentially decreasing learning rate. No dropout, data augmentation or post-processing is used.

Linear regression classifier An algorithm that classifies image patches using linear regression. The features the classifier looks at are the mean and standard deviation of the pixel intensities of the patches.

In order to make meaningful comparisons, we have used the same training, evaluation and test sets for the baseline algorithms and our new algorithm. The stopping condition for the CNN patch classifier implementation is also similar to the stopping condition of our network. Our algorithm performs well compared to the two baseline algorithms and makes the most accurate predictions on the test set.

	LR	CNN patch classifier	Our CNN
F1 Score	0.56720	0.80030	0.93535

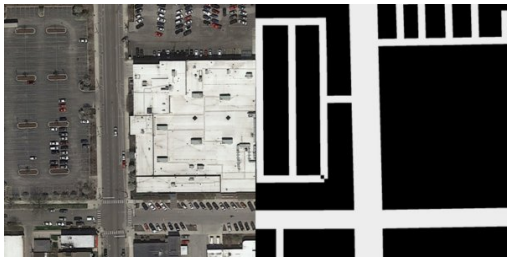

Fig. 10. Example image of a parking and corresponding ground truth from the training set.

Fig. 11. Output of the CNN for a parking in the evaluation set. The ground truth is marked in green, true positives in yellow and false positives are marked red.

IV. Discussion

A. Training data inconsistency

It is interesting to see how our network is able to detect roads in the evaluation set which are not labelled as such in the supplied groundtruth. Figure 10 shows an image of a parking lot from the training set. The roads in the parking are marked here. When the network is run on the evaluation set, it can detect roads in parking lots as seen in Figure 11. They are not labelled as roads here, which leads to false positives. Those inconsistencies in the training and evaluation set leave the network indecisive about certain types of roads.

B. High level road structure

The usage of context information (wide area around the patch) plays an important role in the classification process. In the fragment of a test set image shown in Figure 12, a part of the road is obstructed by trees but the network correctly classifies it as a road as seen in Figure 13. However, other trees are not classified as roads, so the network does not classify trees in general as roads. Therefore, the decision to classify the tree-obstructed region as a road must have been made based on the context and not the region itself.

C. Pre-trained weights

The design of our network is influenced by the configuration of AlexNet [3]. Starting with convolution filters and biases that already represent interesting features instead of taking random starting values is known as pre-training. However, we had to abandon this idea, because our network using the layer sizes of AlexNet did not fit into the available memory.

Fig. 12. Fragment of a test set image, the bigger marked region contains a road obstructed by trees, and the smaller region contains a tree which is not part of a road. The output of the network for this image is shown in Figure 13.

Fig. 13. Input image from Figure 12 with overlaid CNN output marked red. The road in the bottom part of the image is classified correctly despite being obstructed by trees. However, other trees are not mistaken for roads.

D. Test on real world data

We have attempted to use the classifier on aerial images of different cities acquired from Google Maps which to the human observer are similar to the test images from the competition. However, it completely failed to detect the roads present in these examples as nearly all patches were classified as non-road. To achieve practical results, the network should be trained on a bigger and more diverse data set.

E. Further work

The following ways to improve the solution could be investigated:

- Training on a bigger and diverse set of images from different cities may allow the classifier to work on a wider range of input images.
- Extending the network to detect more classes, such as crossings, buildings or rail tracks would increase the amount of information gained from the training set and possibly allow the network to learn the dependencies between those classes, such as that between crossings there is usually a road.

REFERENCES

[1] Amit Bhaya and Eugenius Kaszkurewicz. Steepest descent with momentum for quadratic functions is a version of the conjugate gradient method. *Neural Netw.*, 17(1):65–71, January 2004.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[4] Volodymyr Mnih. *Machine Learning for Aerial Image Labeling*. PhD thesis, University of Toronto, 2013.

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.