Patchwise Road Segmentation for Aerial Images with CNN

Emmanouil Angelis, Spyridon Angelopoulos, Georgios Touloupas
Group 5: Google Maps Team
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—Convolutional networks are powerful visual models that have become the state of the art in the field of image recognition. We propose a Convolutional Neural Network that is based on the VGG architecture and Deep Residual Learning for the task of road segmentation of aerial GoogleMaps images. We experiment with and apply many successful techniques for faster training and regularization, like transfer learning, batch normalization and dropout. The implemented models and methods are evaluated in the Kaggle competition "cil-road-segmentation-2017".**

I. Introduction

Convolutional Neural Networks (CNN) demonstrate outstanding performance in various image recognition problems, ranging from image classification [1] to semantic segmentation [2]. When it comes to the topic of road segmentation, extraction of roads from aerial images has always been an important task.The applications include obstacle and pedestrian detection, road scene understanding and autonomous navigation of cars and drones [3],[4].

In this project we combine several of the successful and recently proposed methods in order to model and train a deep neural network for the task of road segmentation on a given set of aerial images. The efficiency of our methods is evaluated on the performance in the test set of the corresponding Kaggle competition for Computational Intelligence Lab course. In order to train our models, 100 RGB images of 400×400 pixels with their corresponding groundtruth are available. The test set is comprised of 50 RGB images of 608×608 pixels. For each 16×16 patch of the images, we are required to output a discrete label, classifying this patch either as road or background. The metric used for the evaluation of our patch-wise predictions is the F1 score function.

Our model is based on the VGG architecture [1] and also uses an implementation of residual neural networks [5]. Batch normalization [6] and transfer learning are used for faster training while dropout is also considered as a regularization technique. The network is trained with tensorflow [7].

Two different alternatives are considered regarding the computation at the patch-wise level. The first one takes each patch separately, with the input of the network being an extended area having the specific patch at its center. These extended patches provide enough context information for the classification of the central part. The second alternative takes as input the image as a whole, then passing it through convolutional and pooling layers only (thus the network being fully convolutional) and finally producing as output a patchwise prediction for the whole image.

II. Related work

CNNs have been succesfully applied to various semantic segmentation tasks, usually formulated as structured pixel-wise labeling problems. In [8] a fully convolutional approach is used whereas in [9] fully connected CRFs are implemented. In [2] a novel idea is applied, which learns a deconvolutional network on top of the convolutional layers adopted from VGG 16-layer net. A multi-layered deconvolutional network is also implemented in [10], but in a different context; it is used as novel visualization technique that gives insight into the function of intermediate feature layers and has the goal to promote understanding for CNNs.

III. Models and Methods

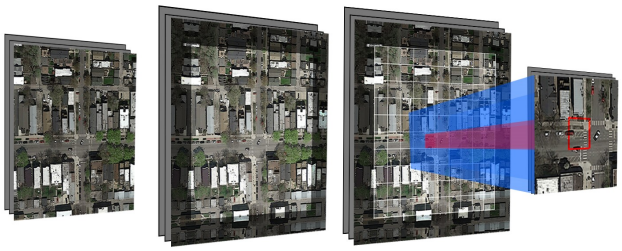

Figure 1. Extraction of extended patches

A. VGG Architecture with Extended Patches

Our first and most successful approach was to implement a CNN based on the VGG architecture. The network up to its fourth pooling layer is identical to the original VGG architecture [1] and has three fully connected layers afterwards as shown in Figure 2. Regarding the input, a subregion of the original image is taken, having at its center the specific patch which we want to classify. We call this subregion an extended patch. This is how we use the context information for patch-wise prediction. Moreover, mirroring at the boundaries of the original image is used in order to avoid black pixels at the extended patches that correspond to the image boundaries. The process of extracting the extended patches is presented in Figure 1.

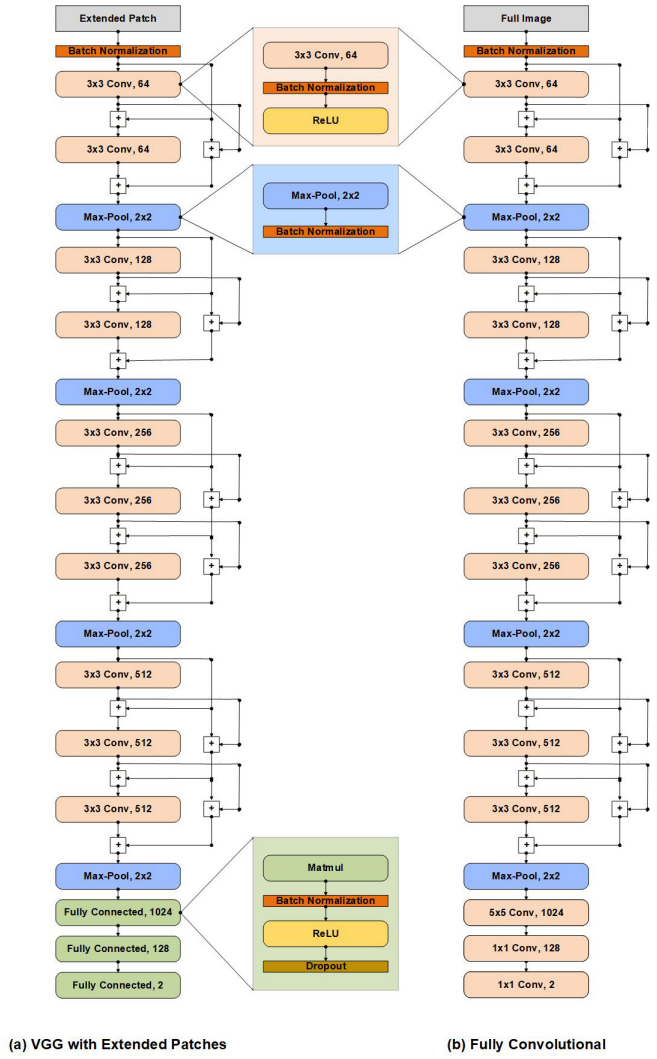

Figure 2. Model architectures

B. Fully Convolutional Model

We also tried to implement a Fully Convolutional Network [8], i.e a model that takes input of arbitrary size and produces correspondingly-sized output by only using convolutional and pooling layers (no fully connected layers are present). In fact, we adapted the previous network (VGG Architecture) as we kept the convolutional and pooling layers and only converted the fully connected layers to convolutional layers which performed similar functionality. After the application of 4 pooling layers to the input image, the spatial size of the image is downsampled by a factor of 16 (the patch size), so every 1×1 spatial position of the output represents a 16×16 patch of the input image.

C. Network Optimizations

1) Pretrained VGG: Transfer learning [11] is a technique commonly used for many visual recognition and semantic segmentation tasks [8], since it can speed up training as well as lead to more accurate results. In our model, we initialize the convolutional layers of the network using a pretrained VGG. Then these layers are not kept fixed, but are fine tuned by continuing the backpropagation.

2) Batch Normalization: Batch normalization (BN) [6] is a method to reduce internal covariate shift in neural networks, potentially accelerating learning and improving performance. A BN layer normalizes (zero mean and unit variance) the output of the previous layer for the current batch before feeding it as input to the next layer. This normalization also acts as a regularizer within the range of the batch.

In our models, BN layers are introduced after the input, convolutional, pooling and fully connected layers, before the non-linearity (ReLU) where applicable.

3) Dense Residual Connections: Residual connections [5] are identity shortcuts that skip intermediate layers and are added to the input of a further layer. They improve the gradient flow during backpropagation, enabling us to train deeper models faster, achieving better results than shallower models. Due to the identity shortcuts it is also more likely for the skipped convolutional layers to learn different filters than the previous layers.

We use dense residual connections [12] for every convolutional block (convolutional layers and pooling layer), where the input of each layer is the sum of the outputs of all previous layers in the block. The output of the layer before the block is also included in the sum, after being projected in order for the dimensions to match.

4) Dropout: A method to prevent the network from overfitting is dropout [13]. During each training step each neuron of a layer with dropout is either kept with probability p or "dropped out" (set to zero) with probability $1-p$. This means that not all neurons are trained on all the data, thus preventing overfitting and speeding up training.

In our extended patches model, dropout is applied to the first two fully connected layers. The convolutional layers have orders of magnitude less parameters than the fully connected layers, therefore we decide not to apply any form of regularization on them.

5) L2 Regularization: Another method to prevent overfitting is L2 regularization. We penalize the magnitude of the squared Euclidian norm of a weight matrix by adding it to the loss objective.

In our extended patches model, L2 regularization is applied on the weights of the fully connected layers. Regularization is not applied on the bias parameters since they do not interact multiplicatively with the data.

6) Early Stopping: To stop training before the network starts to overfit we use an early stopping criterion[14]. The criterion we choose is stopping training after 10 recording steps (each including 1000 training steps) without improvement for the loss and the F1 score on the validation set. The model that achieved either the minimum loss or the maximum F1 score on the validation set is saved to generate

the predicitons on the test set.

D. Postprocessing

1) Model Ensemble: To further improve the performance of the network, we combine our best performing models throughout the experiments with ensemble averaging. The ensemble exhibits reduced variance and average bias compared to the individual models. The increase in performance is bigger if the models are less correlated, thus we choose models trained with different hyperparameters for the ensemble.

The softmax probabilities for all test image patches are stored for each model of the ensemble. These probabilities are averaged among the best performing models to construct the ensemble patch probabilities, which in turn are used to calculate the ensemble predictions.

2) Fully connected CRFs: Conditional Random Fields (CRFs) [15] are graphical models often used in object recognition and image segmentation tasks. CRFs can refine the softmax probabilities of our model for a test image by using the RGB intensities of that image. This is done by minimizing an energy function consisting of a Gaussian and a bilateral pairwise potential. The edges (big intensity changes) in the test image are used to refine the softmax probabilities, while small segmentation regions are penalized.

We use a fully connected CRF model [16] [17] that establishes a Gaussian and a bilateral pairwise potential on all of the pixels of the test image. The softmax probabilities of the model ensemble are computed per patch, therefore we smooth them by applying a Gaussian blurring filter before are using them as input to the CRF.

IV. Experiments

Experiment	F1 Score
BASELINE-LOG	0.56751
BASELINE-CNN	0.82582
BASELINE-CNN-AUG	0.83626
BASELINE-CNN-AUG-45	0.83662
VGG	0.85884
VGG-PRETRAINED	0.86169
VGG-BATCHNORM	0.86654
VGG-RESIDUAL	0.86706
EXTENDED-36	0.91548
EXTENDED-56	0.93086
EXTENDED-76	0.93141
EXTENDED-92	0.93385
EXTENDED-56-DROP-0.5	0.92280
EXTENDED-56-DROP-0.7	0.92512
EXTENDED-56-DROP-0.9	0.92801
FULLY-CONVOLUTIONAL	0.90762
ENSEMBLE	**0.94152**
CRF	0.93266

Table I
F1 scores on the Kaggle competition

A. Baseline Models

1) Logistic Regression: Our first baseline is a logistic regression classifier. To train the classifier two features are used per image patch, the mean and the variance of the gray color. The score achieved by `BASELINE-LOG` (0.56751) as seen in Table I is the lowest among all our experiments. This is expected since the logistic regression classifier is linear on the feature hyperplane and the features used were not sophisticated.

2) Baseline CNN: The second baseline is a rather shallow CNN. This model has the following architecture: CONV32-RELU-POOL-CONV64-RELU-POOL-FC512-RELU-FC2. The only optimizations enabled are L2 regularization on weights of the fully conneted layers and the use of the early stopping criterion described in Subsection III-C6. The model was trained on 16×16 patches from a 90:10 split of training and validation images from the original training dataset of 100 images. `BASELINE-CNN` has considerably better performance than `BASELINE-LOG`, reaching an F1 score of 0.82582. This is a success for the CNN model approach, which we follow for the design of our models.

B. Data Augmentation

As mentioned in the introduction we have at our disposal 100 RGB images of 400×400 pixels with their corresponding groundtruth are available for training. Since the dataset is small we decide to augment it by rotating each image by 90, 180 and 270 degrees and then keeping these images as well as their horizontally flipped counterparts. The corresponding transformations are performed on the groundtruth images as well. The images of the augmented dataset are shuffled and we get the augmented training and validation sets with a 90:10 split. The augmented dataset is 8 times bigger than the original. When training the CNN baseline on the augmented dataset (`BASELINE-CNN-AUG`) we notice the expected increase in performance over the original dataset (`BASELINE-CNN`), as seen in Table I.

We also experimented with further augmenting the dataset by adding 45, 135, 225 and 315 degree rotated and zoomed images. However there was no noticeable increase in performance (`BASELINE-CNN-AUG-45` vs `BASELINE-CNN-AUG` in Table I), therefore we decided to use the first augmented dataset (without 45 degree rotations) for the following experiments.

C. VGG Architecture with Extended Patches

1) Plain VGG: In experiment `VGG` we train the VGG architecture of Figure 2 on non-extended 16×16 patches to compare it with `BASELINE-CNN-AUG`. As expected the VGG model performs noticeably better, with an F1 score of 0.85884.

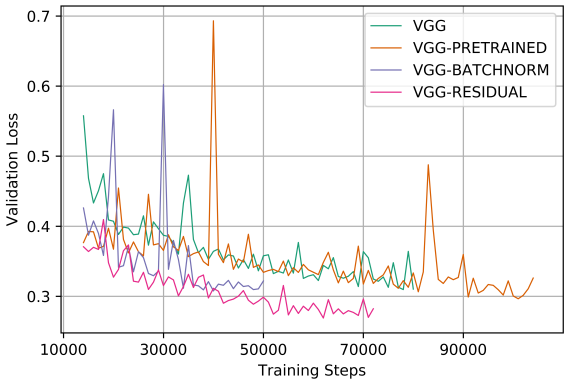

Figure 3. Average loss on validation during training for models trained on 16×16 patches

2) Pretrained VGG: The use of pretrained weights to initialize the convolutional layers of the VGG architecture leads to improved performance for `VGG-PRETRAINED`. In Figure 3 we can also see that `VGG-PRETRAINED` achieves consistently lower loss than `VGG` during the first 30000 steps. For these reasons we chose to use the pretrained VGG weights for all following experiments.

3) Batch Normalization: The use of BN leads to a slight increase in performance as seen in Table I. What is more important is the significantly faster convergence of the `VGG-BATCHNORM` model as seen in Figure 3, which lead us to the adoption of BN for the next models.

4) Dense Residual Connections: Using dense residual connections on the convolutional blocks yielded almost no increase in performance over `VGG-BATCHNORM`. However in Figure 3 we can see that the `VGG-RESIDUAL` model reaches lower validation loss faster than the `VGG-BATCHNORM` model. For this reason we decided to use dense residual connections in the next models.

5) Dropout: We experiment with different values of the keep probability p for dropout on the first two fully connected layers. For the experiments we now use 56×56 patches for training, as this is a good starting point. We use keep probability 50%, 70% and 90%, naming these experiments `EXTENDED-56-DROP-0.X` and name the no dropout experiment `EXTENDED-56` correspondingly. In Table I we can see that as we decrease the dropout (increase the keep probability) the model achieves better performance, therefore we decide against using dropout for the next experiments. This may be attributed to the fact that we already have other forms of regularization on these layers, namely BN and L2 regularization.

6) Extended Patch Size: For the final experiments we tune the size of the extended patch. We experiment with 36×36, 56×56, 76×76 and 92×92 patches, while enabling all the aforementioned optimizations except from dropout. As we can see in Table I the F1 score increases as the extended patch size increases. The limit of 92×92 patches not surpassed due to GPU memory limitations, however the increase in performance after 56×56 was not big.

D. Fully Convolutional Model

The `FULLY-CONVOLUTIONAL` model clearly takes advantage of the overlapping regions that surround neighboring patches, and thus training is faster compared to the model of extended patches. On the other hand, it is not as accurate, as we can see in Table I, achieving an F1 score of only 0.90762.

A possible explanation for this failure is that information coming from far away regions is taken into consideration for the determination of each patch, while in the case of extended patches, only the immediate surroundings play a role.

E. Postprocessing

1) Model Ensemble: To exploit the good performance and variety of the models we have already trained for the previous experiments, we form an ensemble of our 3 best performing models: `EXTENDED-56`, `EXTENDED-76` and `EXTENDED-92`. As expected, the `ENSEMBLE` submission scored an improved F1 score of 0.94152, which is the best score we achieved for the Kaggle competition.

2) Fully Connected CRFs: We experimented with different values for the tuning parameters of the Gaussian and the bilateral pairwise potentials of the CRF energy function, however we did not manage to further improve on the F1 score of the `ENSEMBLE` as it is shown in Table I.

A reason for this is because F1 scores were computed on annotations that were not performed on a view pixel basis. As we can see in Figure 4, there are areas where aerial view of the road is obstructed by trees and cars, and the ideal classifier for this setting would predict these areas as road. However CRFs perform pixelwise classification based on the RGB intensity changes on the test image and therefore classify these pixels as background.

V. Summary

In this project we constructed deep CNNs for the task of road segmentation of aerial images. We mainly experimened with a VGG architecture based on extended patches and a corresponding fully convolutional VGG network that takes the whole image as input. We verified that the first approach produces better results. Dense residual architecture had a beneficial impact on the prediction accuracy and accelerated the training process. When it comes to the optimization techniques used for faster training and regularization, we concluded that indeed the methods of transfer learning and batch normalization improve the overall performance. On the other hand, the dropout method did not result into a higher prediction accuracy. Finally, by selecting an ensemble of our three best models we managed to achieve our best F1 score in the Kaggle competition.

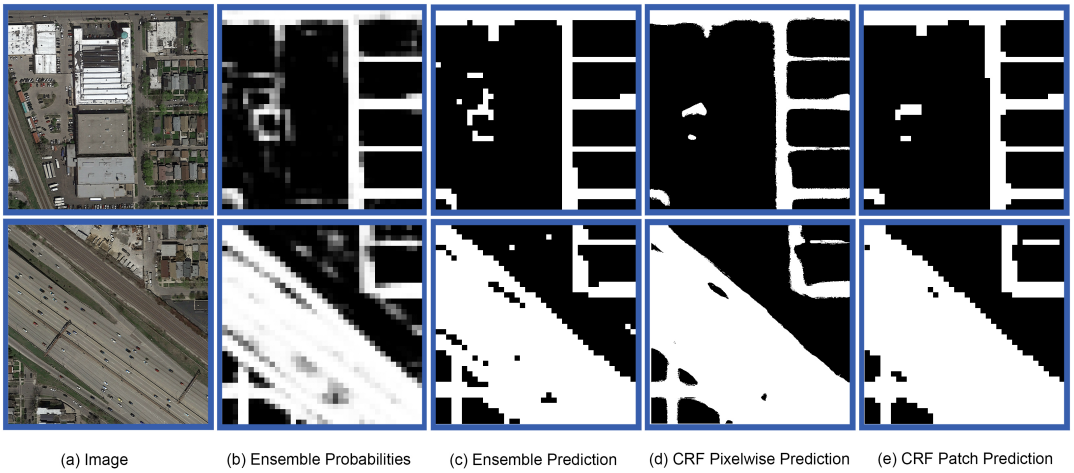

Figure 4. Qualitative results for the model ensemble and the fully connected CRFs

REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[2] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1520–1528.

[3] J. Alvarez, Y. LeCun, T. Gevers, and A. Lopez, "Semantic road segmentation via multi-scale ensembles of learned features," in *Computer Vision–ECCV 2012. Workshops and Demonstrations*. Springer, 2012, pp. 586–595.

[4] I. Laptev, H. Mayer, T. Lindeberg, W. Eckstein, C. Steger, and A. Baumgartner, "Automatic extraction of roads from aerial images based on scale space and snakes," *Machine Vision and Applications*, vol. 12, no. 1, pp. 23–31, 2000.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[8] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *arXiv preprint arXiv:1606.00915*, 2016.

[10] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *arXiv preprint arXiv:1311.2901*, 2013.

[11] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[12] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," *arXiv preprint arXiv:1608.06993*, 2016.

[13] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[14] L. Prechelt, "Early stopping-but when?" *Neural Networks: Tricks of the trade*, pp. 553–553, 1998.

[15] J. Lafferty, A. McCallum, F. Pereira *et al.*, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[16] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *Advances in neural information processing systems*, 2011, pp. 109–117.

[17] "PyDenseCRF," https://github.com/lucasb-eyer/pydensecrf. [Online]. Available: https://github.com/lucasb-eyer/pydensecrf