In Proceedings of the European Symposium on Artificial Neural Networks (ESANN), Bruges, April 2012

Learning Object-Class Segmentation with Convolutional Neural Networks

Hannes Schulz and Sven Behnke

University Bonn – Computer Science VI, Autonomous Intelligent Systems
Friedrich-Ebert-Allee 144, 53113 Bonn – Germany

Abstract. After successes at image classification, segmentation is the next step towards image understanding for neural networks. We propose a convolutional network architecture that includes innovative elements, such as multiple output maps, suitable loss functions, supervised pretraining, multiscale inputs, reused outputs, and pairwise class location filters. Experiments on three data sets show that our method performs on par with current in computer vision methods with regards to accuracy and exceeds them in speed.

1 Introduction

Neural networks have a long history of usage for image classification, e. g. on MNIST, NORB, and Caltech 101. For these datasets, neural networks rank among the top competitors [1]. Despite the success, we should note that these image classification tasks are quite artificial. Typically, it is assumed that the object of interest is centered and at a fixed scale, i. e. that the localization problem has been solved. Natural scenes rarely contain a single object or object class. Object detection and object-class segmentation are thus the logical step towards general image understanding. In this work, we propose variations of the convolutional neural network (CNN) for object-class segmentation. Specifically, we show that CNN can compete with state-of-the art object-class segmentation methods on three common datasets, that supervised pre-training generally improves performance and that during recall, CNN are faster than comparable approaches.

2 Methods

Network Architecture Our network architecture (Figure 1) extends the standard CNN architecture [2]. In addition to the standard architecture, we introduce several notable differences. Firstly, we use multiple maps for output, since we are dealing with image-like outputs and a multi-class classification problem. The cost function of the output map is either the pixel-wise cross entropy loss $E_{\text{ce}}(\mathbf{y}, \mathbf{o}) = -\sum_i y_i \ln \left[\exp(o_i) \,/\, \sum_j \exp(o_j)\right]$ (i. e. softmax) or the pixel-wise squared ε-insensitive loss after a sigmoid non-linearity $E_{\text{ei}}(\mathbf{y}, \mathbf{o}) = \max\left(0, |y_i - (1 + \exp(o_i))^{-1}| - \varepsilon\right)^2$. Here, $\mathbf{y}$ and $\mathbf{o}$ denote teacher and net output, respectively. The choice of the loss function depends on the task: we choose E_{ce} when we are interested in the best prediction per pixel and E_{ei} when classes are evaluated separately. Secondly, we use a pretraining approach with intermediate

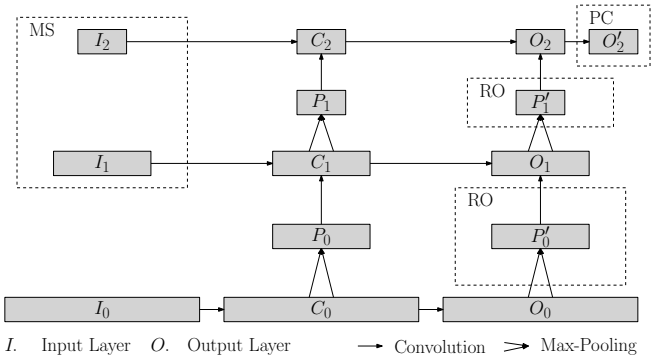

Fig. 1: Network with three convolutional layers. Dashed parts are optional: *MS* multiscale inputs, *RO* reused outputs, *PC* pairwise class location filter.

output layers. In contrast to [3, 4], however, we do not discard pretraining results, but reuse them as input to higher layers (RO in Figure 1). To achieve this, we max-pool output layers and use an identity-initialized convolution to the next output layer. With this, the next layer can focus on learning the difference to the output of the previous layer. Thirdly, inspired by spatial pyramids, we provide inputs at multiple, coarser scales to the network (MS in Figure 1). Finally, it was found that in multi-class settings, long-range dependencies between classes can be exploited [5]. We therefore add a pairwise class location filter (PC in Figure 1) to the output, a convolution with a wide filter learned on top of the output layer.

"Valid" convolutions shrink the image by a margin proportional to filter size. For flexibility, we keep the relative positions of teacher and the output map constant. Therefore, before convolution, we pad each intermediate map with the mean of that map, keeping the map size constant and largely avoiding border effects as a result. Unless stated otherwise, our input resolution is 176×176 and hidden layers have 32 maps. Filters of convolutions are initialized randomly [6]. Intermediate layers C_i apply the non-linearity $\tanh(\cdot)$ followed by a 2×2 maximum pooling P_i.

Preprocessing To facilitate batch processing on GPU, we first scale the image and place it in the center of a fixed-size square image. From the squared RGB images, we extract two kinds of feature, zero phase whitening (ZCA) and histogram of oriented gradients (HOG). A convolutional $5 \times 5 \times 3$ ZCA whitening transform removes first-order correlations between neighboring pixels and between color channels. HOG is computed for a single scale with five non-oriented bins. The resulting eight maps are standardized to zero mean and unit variance separately. Teachers, given as a map of class indices, are transformed into one map per class, which has value one iff the pixel is associated with the class. In addition, we create a "don't care" mask, which is used to eliminate gradients from parts of the square image which do not belong to the original image or are not labeled in ground truth.

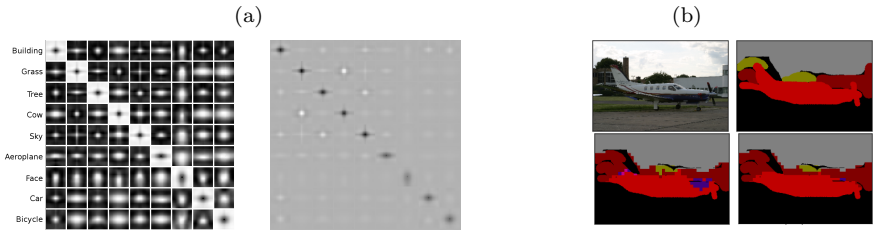

Fig. 2: (a) Pairwise class location filters (PC) learned for MSRC-9. Black represents more positive weights. Each prior is normalized separately (right: all normalized together). (b) PC repairs spurious observations. Top row: original and ground truth; bottom row: before and after PC.

Learning We use batch stochastic gradient descent with a learning rate of $\eta_T = 10^{-3} \cdot 0.97^T$, where T denotes the epoch number. In pooling layers, the error is propagated only to the location of the maximum—for details, see [7].

Similar to previous work [3, 8], we use supervised pre-training. Starting with O_0, each output layer O_i is trained for 50 epochs. Afterwards, the loss at O_i is fixed to zero, O_i becomes a regular hidden layer, and O_{i+1} is trained. Note that the derivative of output layers with softmax $S(\mathbf{x})$ simplifies considerably when used in combination with cross-entropy loss. Here, we use softmax nonlinearities in hidden layers and therefore have to use the more involved $\partial S_i(\mathbf{x})/\partial x_j = S_i(\mathbf{x})(\delta_{ij} - S_j(\mathbf{x}))$.

All operations are performed on GPU using the CUV[1] library [9].

3 Results

To measure final outcome, we crop the region of the original image from the (quadratic) output maps and scale up to the original image size.

We evaluate our architecture on three datasets. MSRC-9 [10] is a 9 class, 240 images dataset with about 70% of the pixels labeled. We split the dataset into a stratified training and test set containing 50% of the images each. MSRC-21 is an extended version of MSRC-9 containing 591 images with 21 labeled classes. We again use the standard split into training, validation and test set [10]. The common evaluation criterion for both is pixel-wise accuracy in labeled regions, we consequently use $E_{\text{ce}}(\cdot,\cdot)$. Finally, we evaluate on INRIA Graz-02 (IG02, [11]). The dataset contains three classes in 479 training and 479 test images. The evaluation criterion here is per-class precision/recall at equal error rate (PR-EER), therefore we use $E_{\text{ei}}(\cdot,\cdot)$. We augment all training sets with horizontally flipped versions of the originals.

Table 1a summarizes our results for MSRC-9. We find that we perform best when using RO+MS+PC, with 90.2% accuracy, improving on the previous result of Grangier et al. [4] by 1.7%. Figure 2a shows the learned pairwise class location filter for MSRC-9. It learned e. g., that "aeroplanes" are horizontally extended objects and that to the left and right of cows, the probability for "building" is

[1]Publicly available at `https://github.com/deeplearningais/CUV`

(a) MSRC-9

Condition	Accuracy
RO+PC+MS	**90.2**
RO+PC	89.8
regular	89.1
no pretraining	87.9
Grangier et al. [4]	88.5

(b) MSRC-21

Condition	RO	RO+PC
1 Layer	45.4	49.4
2 Layer	71.0	76.9
3 Layer	76.0	77.7
3 Layer+MS	77.9	80.2
4 Layer	77.4	77.8
Gould et al. [5]	70.1	77.8
Ladicky et al. [13]	**86.0**	

Table 1: Segmentation results for the MSRC datasets

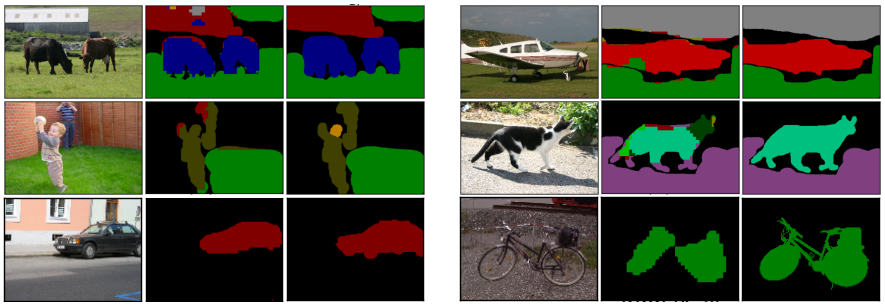

Fig. 3: Example segmentations for MSRC-9 (top row), MSRC-21 (center row), IG02 (bottom row). Left column shows original image, center column our output, right column ground truth. MSRC has ground truth "void" mask superimposed.

low. As demonstrated in Figure 2b, this filter helps to remove spurious detections where the classifier was unsure. Further example segmentations are displayed in Figure 3.

For MSRC-21, we analyze the advantages of the pairwise class location filter as we vary the number of layers (Table 1b). We find that PC improves the result of the lower layers. The effect diminishes, however, when resolution becomes too small (22×22 for 4 Layers). While our best result (80.2%) does not achieve state-of-the art, it improves upon Gould et al. [5] who proposed the relative location prior which inspired PC. Without pretraining, we get only 73.9%, which emphasizes its importance.

On IG02, we find that PC does not help as much (Table 2). We attribute this to the structure of the dataset (one class per image, few classes in total). Using RO only, we can already improve upon the previously best result [12] in two out of three classes.

On a NVIDIA GTX 580, we process 16 images at once. A batch forward pass through the best-performing network on IG02 takes 0.23 s, which amounts to approx. 70 images per second. We are currently rewriting the preprocessing to work online. It can be executed concurrently to the forward pass on CPU. Our

Condition	Car	Bike	Person
RO	**75.6**	74.7	61.1
RO+PC	**75.6**	**74.8**	61.1
regular	74.4	74.2	61.9
Fulkerson et al. [12]	72.2	72.2	**66.3**
LIN[14]	62.9	71.9	58.6

Table 2: INRIA Graz02 results

naïve HOG implementation takes 0.04 s/image at input resolution. While the ZCA whitening transform amounts to a convolution and therefore takes less time than the forward pass. Consequently, we expect framerates of > 10 fps, which is an order of magnitude faster than e. g. speed-optimized work by Aldavert et al. [14] (cf. their results in Table 2).

4 Related Work

While a large body of research focuses on object-class segmentation, most notably associated with the Pascal VOC challenge[2], only few works have attempted to use a neural network architecture similar to ours.

The first attempt in the direction is made by Jain and Seung [3], who use supervised pre-training for a denoising task and show that their approach has relations to the optimization of a Markov random field. This work differs from ours in the task (regression vs. classification), and in architecture, since the authors do not reuse previous outputs.

Mnih and Hinton [8] use unsupervised pre-training for a network that classifies roads in aerial images. Their dataset is not publicly available and the classification problem is binary. The authors use methods from Jain and Seung for post-processing only, while we continue training through the network.

Both, [3] and [8], train on image patches. We use convolutions instead, which are faster since intermediate results are shared between adjacent locations.

Gould et al. [5] introduced the idea of a relative location prior in conditional random fields, addressing the problem that certain classes have different probabilities of co-occurrence depending on their relative spatial location. We draw on their idea by explicitly learning filters that reduce errors of previous class predictions.

Finally, Grangier et al. [4] used an architecture similar to ours. From their brief description, many details of the algorithm remain unclear. In contrast to their approach, we reuse the pretraining results. We directly compare our results on MSRC-9 with theirs.

In contrast to most neural networks for image processing, we use densely extracted image features to complement raw color channels.

[2] `http://pascallin.ecs.soton.ac.uk/challenges/VOC/`

5 Conclusion

We presented a convolutional neural network architecture for object-class segmentation, which achieves state-of-the-art performance on common vision datasets. Crucial factors for good performance are supervised pretraining and pairwise class location filters. The network can be parallelized well on GPU and exhibits very good recall times. The outputs of our network still do not look visually pleasing. To this end, we are currently experimenting with conditional random fields to make class boundaries respect image edges.

References

[1] D. Ciresan, U. Meier, J. Masci, L. Gambardella, and J. Schmidhuber. "High-Performance Neural Networks for Visual Object Classification". In: *CoRR* abs/1102.0183 (2011).

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* 86.11 (1998).

[3] V. Jain and H. Seung. "Natural image denoising with convolutional networks". In: *NIPS 21*. 2008.

[4] D. Grangier, L. Bottou, and R. Collobert. "Deep convolutional networks for scene parsing". In: *ICML 2009 Deep Learning Workshop*. 2009.

[5] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller. "Multi-class segmentation with relative location prior". In: *Computer Vision* 80.3 (2008), pp. 300–316.

[6] D. Erhan, P. Manzagol, Y. Bengio, S. Bengio, and P. Vincent. "The difficulty of training deep architectures and the effect of unsupervised pre-training". In: *AISTATS*. 2009, pp. 153–160.

[7] D. Scherer, A. Müller, and S. Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition". In: *ICANN*. 2010, pp. 92–101.

[8] V. Mnih and G. E. Hinton. "Learning to Detect Roads in High-Resolution Aerial Images." In: *ECCV*. 2010, pp. 210–223.

[9] H. Schulz, A. Müller, and S. Behnke. "Exploiting local structure in Boltzmann machines". In: *Neurocomputing* 74.9 (2011), pp. 1411–1417.

[10] J. Shotton, J. Winn, C. Rother, and A. Criminisi. "Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation". In: *ECCV* (2006), pp. 1–15.

[11] M. Marszatek and C. Schmid. "Accurate object localization with shape masks". In: *CVPR*. 2007, pp. 1–8.

[12] B. Fulkerson, A. Vedaldi, and S. Soatto. "Class segmentation and object localization with superpixel neighborhoods". In: *ICCV*. 2009, pp. 670–677.

[13] L. Ladicky, C. Russell, P. Kohli, and P. Torr. "Associative hierarchical crfs for object class image segmentation". In: *ICCV*. 2009, pp. 739–746.

[14] D Aldavert, R. De Mantaras, A. Ramisa, and R. Toledo. "Fast and robust object segmentation with the Integral Linear Classifier". In: *CVPR*. 2010, pp. 1046–1053.