A Convolutional Neural Network for Blur Segmentation

Visual Computing Lab

René Schuster

31.03.2016

Abstract In this computer vision lab, we propose a deep learning approach for blur segmentation. We use a neural network that has previously been proposed for motion field estimation with two convolution layers. We do patch based training with an existing blur detection dataset as well as with artificially generated images. We compare our method with previous approaches and show that we score slightly better results.

1 Introduction

Blur is an effect in imaging or photography that reduces the contrast or sharpness and thus the level of detail. Blur is commonly divided into motion blur and out-of-focus blur. Motion blur originates from motion of the camera or of an object during exposure time. It is often assumed as uniform linear motion but occurs also non-uniformly and in combination with rotational motion. Out-of-focus blur is an optical aberration that blurs the regions of an image that are outside of the field of depth. Both types of blur can degrade image quality or serve as a visual effect in photography or photo editing. Blur has direct impact on the frequency of images, e.g. out-of-focus blur can be interpreted as a low pass filter.

Blur segmentation is a task in Computer Vision with the goal of splitting an image into sharp and blurry regions. It can be the basis for many applications like non-blind deblurring, blur enhancement, image segmentation, depth and motion estimation, object detection or a measure for image quality.

With the recent second renaissance of neural networks, deep learning methods are rapidly conquering every field of Computer Vision and sometimes perform even better than the human visual system. Therefore, we are tackling the problem of blur segmentation with a convolutional neural network. We will use a network architecture that has already been proposed for a closely related vision task and incorporate slight improvements and adjustments. Apart from an existing image dataset, we will come up with a blur generation model, to provide our network with more data.

2 Related Work

Before people had started to do blur segmentation or classification, there has been research on the general blur extent of images [4, 14]. While some use Wavelet transformation to determine if an image is blurred and by which extent [14], others use a gradient based approach to also tell whether an image is completely blurred or only partially, and of what type the primary blur is [4].

Others have focused on only one type of blur. Given motion kernel candidates, the method in [2] locally chooses the best kernel to obtain a motion blur segmentation. In practice this method is quite limited due to the restriction to vertical and horizontal motion kernels. Though to some degree it is also applicable for out-of-focus blur. There are also methods specialized on the latter type [15, 10]. Using the local extent of defocus blur, the approach in [15] obtains a dense estimation that is comparable to a depth map and achieves good results in segmentation of foreground and background. The analysis of barely visible focal blur in [10] has led to a versatile approach with very good results in blur segmentation, refocus and depth estimation.

The method in [6] is based on 4 local features to determine the blur and its type for each pixel. Based on that, the authors of [9] used similar features and added a set of learned local filters. Comparable, but with only one feature each for detection and classification, the method in [12] uses singular value decomposition and an alpha channel model.

In summary, there is quite a variety of feature based approaches either in frequency domain or based on image statistics primarily exploiting gradient properties of sharp and blurry images or combining several features.

Contrary to previous work, we use an artificial neural network to learn both, features and classifier. Our network topology is very close to the proposed design in [13], in which the authors estimate a non-uniform motion field for deblurring. We want to train our network on both types of blur with the exclusive goal of blur segmentation.

3 Network Architecture

We solve the problem of blur segmentation as a binary classification problem using an artificial neural network with 7 layers. The network topology consists of two convolution layers each combined with a max-pooling layer and followed by two fully connected layers with a final softmax layer. The network architecture is visualized in Figure 1.

The first convolution layer defines 96 kernels of size 7x7 with 3 color channels which are densely applied to the input patch. For both convolution layers and the first fully connected layer, we use the Rectified Linear Unit (ReLU) as activation function, that has first been proposed in [8]. There are two major advantages of ReLU. First, it is less likely that the gradient vanishes, since the function is constant for input greater than zero. Secondly, it enforces sparsity in the network, since the function is zero for all input smaller than zero. The learned kernels of this convolution layer are shown in figure 3. Both max-pooling layers share the same options: A square kernel of size 2 with a stride of 2, i.e. the pooling regions do not overlap. The second convolution layer creates 256 filters of size 5x5 with 96 channels. Again the kernels are applied completely overlapping, i.e. the stride is 1. For the larger first fully connected layer, we also apply a dropout layer with a ratio of 50 % as in [11] to avoid over fitting. Dropout randomly sets the given percentage of nodes in a layer to zero. It can be imagined as the sub-sampling of a network into many smaller networks. This way, the output can not rely on any smaller part of the network only, which enforces better generalization. The first fully connected layer has 1024 output nodes and over 6 million weights, so using dropout is useful, especially for higher learning rates. The second inner-product layer lowers the dimension down to two nodes and uses no activation function. The last layer is a softmax layer, where we treat the output values as probabilities for the 2 classes *sharp* and *blurred*. For the center pixel of each patch we choose the class that is more probable.

The concrete differences between our topology and the one in [13] are as follows

- we use an odd image patch size of 31 instead of 30.
- we have attached a dropout layer to the large inner product layer.
- our output has only 2 nodes, not 73, since we only do binary classification.

Using the previously presented network, we have done several large data experiments.

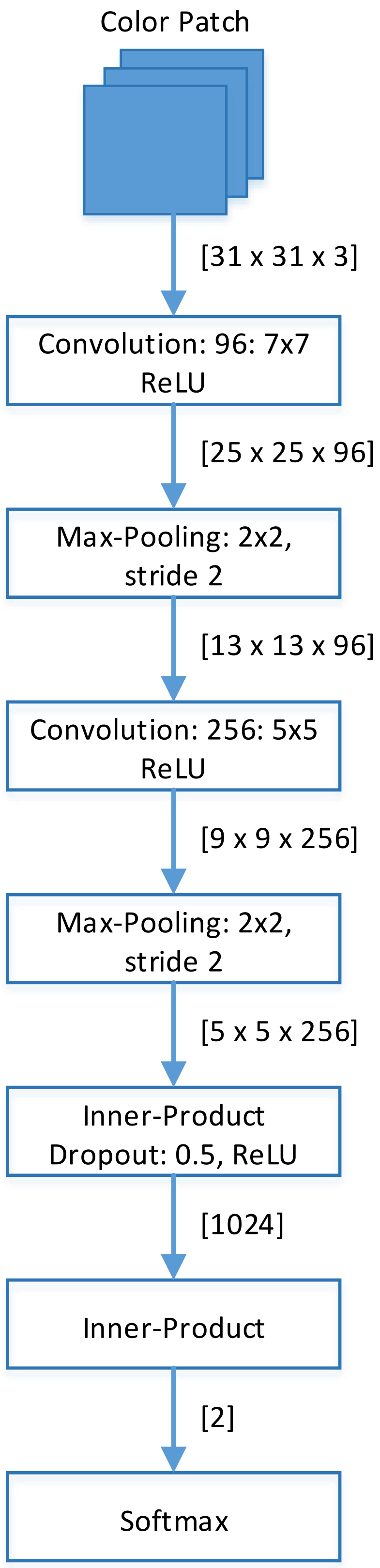
Color Patch
[31 x 31 x 3]
Convolution: 96: 7x7 ReLU
[25 x 25 x 96]
Max-Pooling: 2x2, stride 2
[13 x 13 x 96]
Convolution: 256: 5x5 ReLU
[9 x 9 x 256]
Max-Pooling: 2x2, stride 2
[5 x 5 x 256]
Inner-Product Dropout: 0.5, ReLU
[1024]
Inner-Product
[2]
Softmax

Figure 1: Network topology of the proposed convolutional neural network. The data dimensions are given inside the brackets next to each transition array.

4 Experiments

The following subsections describe the variety of experiments that we have done. First, we show how the training data is provided. After that, we give details about the implementation and training process. Finally, we measure the accuracy of our approach and compare it to other work.

Training data. To train our network properly we need a huge amount of sharp and blurred image patches with ground truth information. We have sampled our image patches from the blur detection dataset created in [9]. This database contains 296 motion images and 704 images with defocus blur of an median size of 640x450. It also supplies a binary segmentation for each image that annotates the regions that are affected by the given blur type.

In order to keep our data unbiased, we have extracted an equal amount of patches from motion and out-of-focus blur images. We also balance the amount of completely sharp, totally blurry and mixed patches, i.e. patches that contain sharp and blurred regions. As a result, the total count of both patch classes are approximately even, if the images contain large enough areas to sample from. After choosing 20 images as evaluation set, we split the remaining images of each blur category into sets for training and validation by a ratio of 4:1, and extract patches of size 31x31. This way, we generate over one million patches for training and around 0.25 millions to keep track of the accuracy during training. From the numbers above, one can see, that the sampled patches share same image areas. We expect that closely overlapping patches with sharp and blurry regions improve spatial precision of the classification.

To obtain a more distinct dataset we have created artificially blurred images for both blur types. This is done using images from the PASCAL Visual Object Class (VOC) dataset [3]. For both, motion and defocus blur, we first randomly choose one foreground image F along its segmentation and a background image B and crop them to a common size. We also select a class from the segmentation of the foreground image that will be used as object mask m. The composition of a motion blurred image is as follows:

$$I_m = k * (m \odot F) + (1 - k * m) \odot B \qquad (1)$$

This is according to the alpha matting model in [5], where I_m is the motion blurred image and k is the motion blur kernel. This model is correct under the assumptions of static background and camera [7]. For our images, we generate randomly oriented linear motion kernels with a length between 5 to 15 and use the motion blurred object mask as ground truth label image.

Similar, we model out-of-focus blur as

$$I_f = (k_2 * m) \odot F + (1 - k_2 * m) \odot (k_1 * B) \qquad (2)$$

where I_f is the defocus blurred image, k_1 is a Gaussian blur kernel to smooth the background and k_2 is a small Gaussian kernel to smooth the transition between foreground and background. This equation is close to the proposed defocus model in [7]. We use a random standard deviation between 1 and 2 to blur the background and the plain object mask as ground truth labeling.

We synthesize 1000 images for each model and create training and validation patches with the same method as for the blur detection dataset afterwards. Two example images are shown in Figure 2. To to be able to compare the quality of our artificial blurred images, our experiments are done on both datasets.

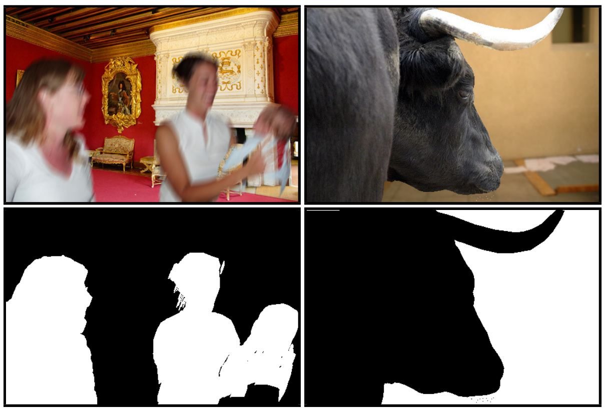
(a) Artificial motion blur image with ground truth (b) Artificial defocus blur image with ground truth

Figure 2: Examples of the artificial blur dataset.

Implementation and Training. The implementation of the network is done in Mocha. A deep learning framework for Julia [1].

We then do one million iterations of stochastic gradient descent with a batchsize of 64, a constant learning rate of 0.0001, a fixed momentum of 0.9, and a regularization (weight decay) of 0.0005 using the CUDA backend on a GeForce Titan X. The learned filters of the first convolutional layer are shown in Figure 3.

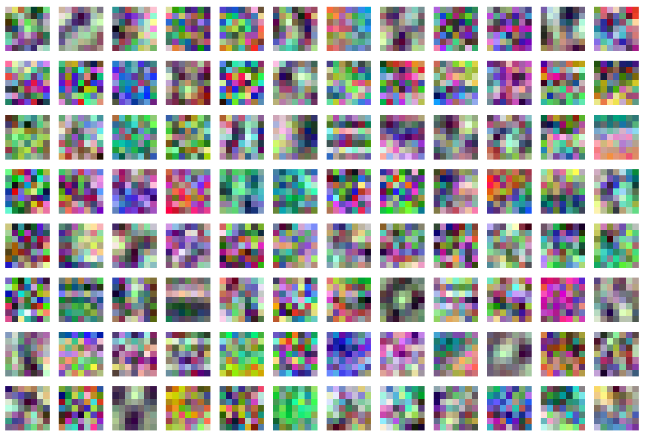
Figure 3: The 96 learned filters of the first convolution layer.

Along the set of kernels in Figure 3, one can see coarse structures, mainly comparable to derivative filters for multiple directions, as well as higher frequency filters, that remotely remind of Garbor or Haar wavelets. This is not surprising, because those filters are primarily used in feature based approaches [6, 9, 14].

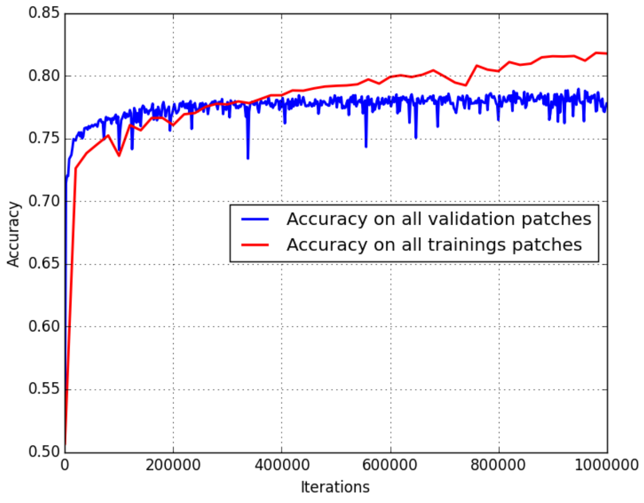

Figure 4: Classification accuracy for training and validation patches over the iterations.

We have also experimented with smaller or higher learning rates, but found this one to suit best. A plot of the patch-wise classification accuracy is shown in Figure 4. It indicates, that we have done fairly enough iterations to reach approximate convergence while we maintain good generalization, i.e. the accuracy on the validation keeps a constant level although the accuracy on the training data increases continuously. Regardless of the fixed learning rate, the last 700000 iterations only have minimal effect on the validation accuracy, which suggest to reduce the total count of iterations in practice. We can not explain why the validation accuracy exceeds the training accuracy at the beginning of the training.

During another series of experiments, we have tested the effect of different input patch sizes. While a larger patch size increases network size and training effort, it does not significantly improve accuracy. Smaller patches dramatically lower the available information and yield no plausible results. The impact of gray scale input data compared to colored images is minor. Also using the gradient magnitude or two channels for x and y derivatives as input lowers the overall accuracy slightly. Using color patches instead of gray scale or gradient input has advantages and disadvantages. Color can be a good blur cue, since the background of natural images is often blue or green or gray and humans are often in the foreground or in motion, but this information also offers potential for overfitting.

Accuracy. To see how capable of handling both blur types our network is, we have trained it using only one type of blur at a time. An overview of this is presented in Table 1.

The accuracy is the average Intersection-over-union of 20 randomly selected and densely classified images for which we have ground truth information. On the one hand it is striking, that motion blur detection in general, and for our artificial training data in particular, performs worse than defocus blur detection. This is a very surprising observation, since the original purpose of the network was motion field estimation. The slightly lower accuracy of the motion network trained with the blur detection dataset can be due to the fact, that this image database contains much less motion images than out-of-focus images. On the other hand, the synthesized out-of-focus blur images deliver an outstanding training result for general images.

Blur \ Data	Blur dataset [9]	Artificial blur
Both	80 %	62 %
Motion	78 %	42 %
Focus	80 %	89 %

Table 1: Accuracy for different input data and blur types.

We have also done an evaluation on the original network in [13], where we use the intermediate result of the estimated motion field to create a motion blur segmentation. This is simply done by thresh-holding the motion vector magnitude at a certain level. Doing so, we reach an average accuracy of 52 % for motion blurred images. One example is shown in Figure 5.

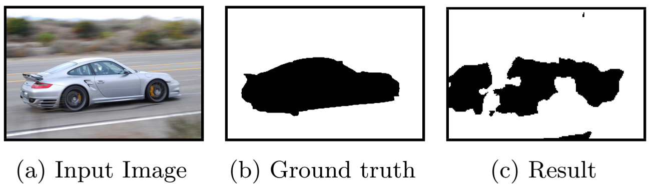

Figure 5: Segmentation result after thresh-holding the motion field of [13].

Finally we have compared our results to state of the art techniques in blur segmentation. To get a feeling for the segmentation results, there is a visual comparison in Figure 7. The quantitative results are shown in Table 2. To produce these results we have used the other authors original published code. The overall accuracy for all methods is the average Intersection-over-Union for 20 blurred images, as before. Ten of those 20 images contain motion blur and ten show out-of-focus blurred regions. All 20 images are densely classified and compared to the ground truth. Our deep learning approach beats previous work on average, although there are some poor results like the one displayed in Figure 6,

5 Conclusion

We have successfully adapted a network for motion field estimation to the task of blur segmentation. In extensive

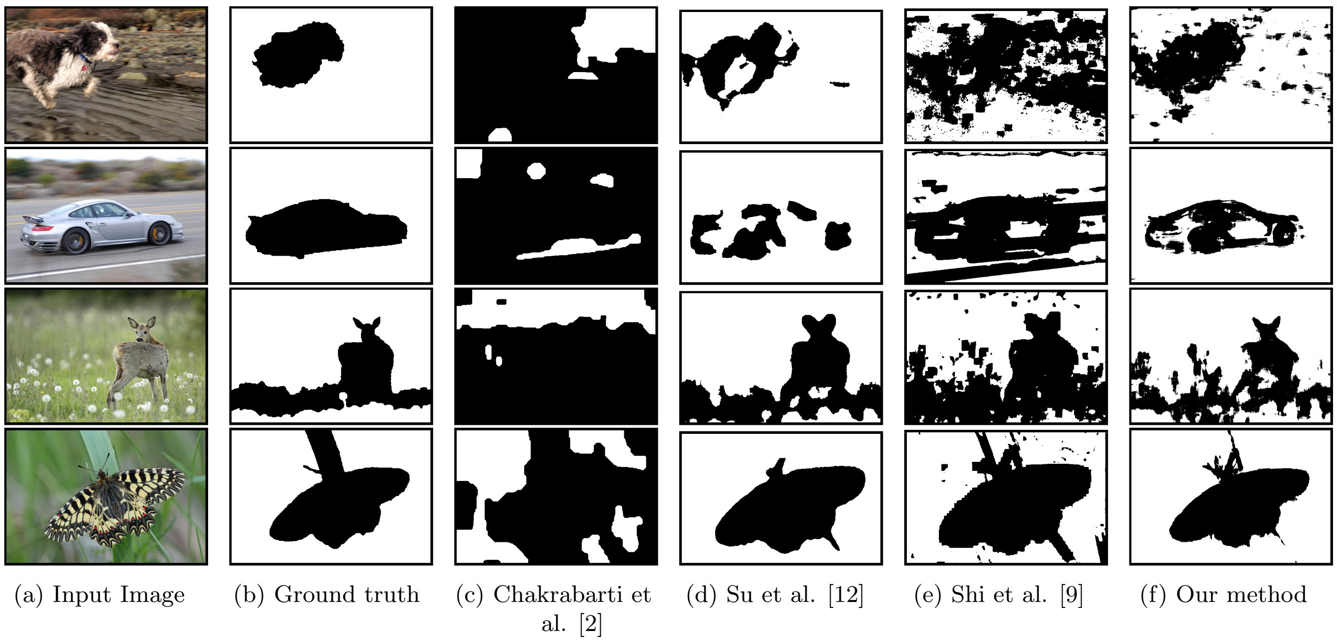
(a) Input Image (b) Ground truth (c) Chakrabarti et al. [2] (d) Su et al. [12] (e) Shi et al. [9] (f) Our method

Figure 7: Qualitative comparison of blur segmentation results.

Method	Average accuracy
Chakrabarti et al. [2]	31 %
Shi et al. [9]	49 %
Su et al. [12]	76 %
Our method	**80 %**

Table 2: Average accuracy for different methods.

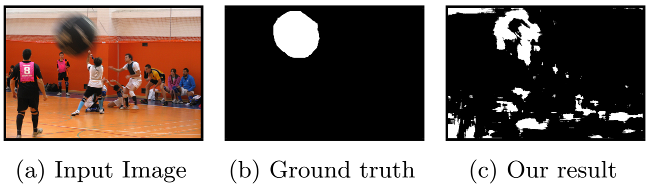
(a) Input Image (b) Ground truth (c) Our result

Figure 6: Negative segmentation example.

experiments, we have shown that our method works for motion and defocus blur and its results can keep up with state-of-the-art approaches. Our artificial generated out-of-focus blur images produce even better results.

The largest drawback of the described method is possibly the ground truth labeling, since both used datasets only annotate one blur type for each image. But many images contain both types.

Therefore in the future we are interested in the creation of more expressive training data to train an improved multiclass network for blur detection. Since the output of the network is a pixel-wise probability, we can also imagine to implement a post processing routine based on a graphical model to smooth the segmentation.

References

[1] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *arXiv preprint (arXiv:1411.1607)*, 2014.

[2] Ayan Chakrabarti, Todd Zickler, and William T Freeman. Analyzing spatially-varying blur. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2512–2519. IEEE, 2010.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[4] Ping Hsu and Bing-Yu Chen. Blurred image detection and classification. In *Advances in Multimedia Modeling*, pages 277–286. Springer, 2008.

[5] Ralph Köhler, Michele Hirsch, Bernhard Scholkopf, and Stefan Harmeling. Improving alpha matting and motion blurred foreground estimation. In *IEEE International Conference on Image Processing (ICIP)*, pages 3446–3450. IEEE, 2013.

[6] Renting Liu, Zhaorong Li, and Jiaya Jia. Image partial blur detection and classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008.

[7] Morgan McGuire, Wojciech Matusik, Hanspeter Pfister, John F. Hughes, and Frédo Durand. Defocus video matting. *ACM Transactions on Graphics*, 24(3):567–576, 2005.

[8] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

[9] Jianping Shi, Li Xu, and Jiaya Jia. Discriminative blur detection features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2965–2972. IEEE, 2014.

[10] Jianping Shi, Li Xu, and Jiaya Jia. Just noticeable defocus blur detection and estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 657–665. IEEE, 2015.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[12] Bolan Su, Shijian Lu, and Chew Lim Tan. Blurred image region detection and classification. In *Proceedings of the 19th ACM international conference on Multimedia (ACMMM)*, pages 1397–1400. ACM, 2011.

[13] Jian Sun, Wenfei Cao, Zongben Xu, and Jean Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 769–777. IEEE, 2015.

[14] Hanghang Tong, Mingjing Li, Hongjiang Zhang, and Changshui Zhang. Blur detection for digital images using wavelet transform. In *IEEE International Conference on Multimedia and Expo (ICME)*, volume 1, pages 17–20. IEEE, 2004.

[15] Shaojie Zhuo and Terence Sim. Defocus map estimation from a single image. *Pattern Recognition*, 44(9):1852–1858, 2011.