Deep Defocus Map Estimation using Domain Adaptation: Supplementary Material

Junyong Lee[1] POSTECH
Sungkil Lee[2] Sungkyunkwan University
Sunghyun Cho[3] DGIST
Seungyong Lee[1] POSTECH

[1]{junyonglee, leesy}@postech.ac.kr [2]sungkil@skku.edu [3]scho@dgist.ac.kr

Abstract

This supplementary document contains more details not shown in the main paper. Sec. 1 first describes more details of the architecture of each subnetwork in DMENet*. In Sec. 2, we report further results evaluated on the sharpness calibration network. Sec. 3 shows the quantitative results of our ablation study that compares different combinations of subnetworks of* DMENet*. Finally, we present more qualitative results on RTF dataset in Sec. 4.*

1. Detailed Network Architecture

Our network consists of four subnetworks: blur estimation network $\boldsymbol{B}$, domain adaptation network $\boldsymbol{D}$, content preservation network $\boldsymbol{C}$, and sharpness calibration network $\boldsymbol{S}$. We here present more details of the structures of the subnetworks.

Blur Estimation Network The main structure of our network $\boldsymbol{B}$ follows U-net architecture [4] with minor changes. The network consists of encoder, decoder, auxiliary module, and residual convolution modules. The encoder directly imports the structure of VGG19 [6]. The decoder adopts the decoder of U-net, in which the up-sampled features are skip-connected with the down-sampled features of the encoder. In the auxiliary module, we attach two consecutive convolutional layers after each up-sampling layer in the decoder. Finally, we attach seven residual blocks of convolutional layers after the decoder, and generate the final result. The details of the network design is shown in Table 1.

Domain Adaptation Network The network $\boldsymbol{D}$ consists of four convolutional layers, followed by the batch normalization layer (see Table 2 for details).

Content Preservation Network Our network $\boldsymbol{C}$ uses the pretrained VGG19 [6].

Sharpness Calibration Network The network $\boldsymbol{S}$ consists of four 1×1 convolution layers. The first three layers are followed by the batch normalization layer. See Table 3 for the detailed architecture.

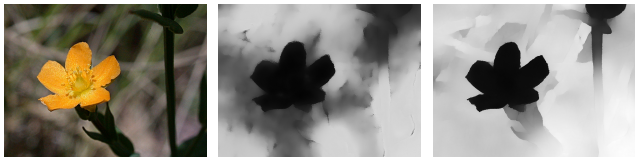

Figure 1: Defocus maps generated with different convolutional filter sizes in the sharpness calibration network $\boldsymbol{S}$: 3×3 filter (middle) and 1×1 filter (right), given an input image (left). The larger filter leads to a smudged defocus map.

2. Evaluation on Sharpness Calibration

This section reports further evaluations performed regarding a larger kernel size and binarization for the sharpness calibration network $\boldsymbol{S}$.

Larger Kernel The filter size of the convolutional layers is 1×1 in the network $\boldsymbol{S}$. We can use a larger kernel size, but it might degrade the accuracy of a generated defocus map. As mentioned in the main paper, with a larger kernel, the receptive field of $\boldsymbol{S}$ becomes larger, and then gradients passed from $\boldsymbol{S}$ to the blur estimation network $\boldsymbol{B}$ would be propagated to larger regions than the receptive fields of $\boldsymbol{B}$. Fig. 1 shows an example of a degraded defocus map.

Binarization One concern may arise is that the network $\boldsymbol{S}$ may *binarize* a defocus map generated by the network $\boldsymbol{B}$. The sharpness calibration loss (L_S) would try to binarize the result if it is directly applied to network $\boldsymbol{B}$. However, we attach network $\boldsymbol{S}$ to $\boldsymbol{B}$, making the gradients of loss L_S to be flexibly applied to $\boldsymbol{B}$. For example, assume that $\boldsymbol{B}$ has estimated a perfect defocus map. If L_S is directly applied to $\boldsymbol{B}$, L_S would be non-zero and influence $\boldsymbol{B}$ towards a binary map. On the other hand, in our case, L_S is applied to $\boldsymbol{S}$ that is attached to $\boldsymbol{B}$, and L_S will be non-zero only when $\boldsymbol{S}$ does not produce a relevant binary blur map. Consequently, $\boldsymbol{S}$

layer type(#)	size	stride	out	act.	repeat
Encoder (VGG19)					
*Conv*1	3×3	(1, 1)	64	*relu*	$\times 2$
maxPool	2×2	(2, 2)	-	-	
*Conv*2	3×3	(1, 1)	128	*relu*	$\times 2$
maxPool	2×2	(2, 2)	-	-	
*Conv*3	3×3	(1, 1)	256	*relu*	$\times 4$
maxPool	2×2	(2, 2)	-	-	
*Conv*4	3×3	(1, 1)	512	*relu*	$\times 4$
maxPool	2×2	(2, 2)	-	-	
*Conv*5	3×3	(1, 1)	512	*relu*	$\times 4$
Decoder					
concat	*upsample*(*Conv*5), *Conv*4				
Conv	3×3	(1, 1)	256	-	$\times 3$
*BatchNorm*1	-	-	-	*lrelu*	
concat	*upsample*(*BatchNorm*1), *Conv*3				
Conv	3×3	(1, 1)	128	-	$\times 3$
*BatchNorm*2	-	-	-	*lrelu*	
concat	*upsample*(*BatchNorm*2), *Conv*2				
Conv	3×3	(1, 1)	64	-	$\times 3$
*BatchNorm*3	-	-	-	*lrelu*	
concat	*upsample*(*BatchNorm*3), *Conv*1				
Conv	3×3	(1, 1)	64	-	$\times 1$
*BatchNorm*4	-	-	-	*lrelu*	
Residual Blocks$\times 7$					
skip	previous layer				
Conv	3×3	(1, 1)	64	-	$\times 2$
BatchNorm	-	-	-	*lrelu*	
add	*BatchNorm*, *skip*				
Auxiliary Modules					
input	*Conv*5				
Conv	3×3	(1, 1)	256	-	$\times 1$
BatchNorm	-	-	-	*lrelu*	
Conv	3×3	(1, 1)	1	-	
input	*BatchNorm*1				
Conv	3×3	(1, 1)	128	-	$\times 1$
BatchNorm	-	-	-	*lrelu*	
Conv	3×3	(1, 1)	1	-	
input	*BatchNorm*2				
Conv	3×3	(1, 1)	64	-	$\times 1$
BatchNorm	-	-	-	*lrelu*	
Conv	3×3	(1, 1)	1	-	
input	*BatchNorm*3				
Conv	3×3	(1, 1)	32	-	$\times 1$
BatchNorm	-	-	-	*lrelu*	
Conv	3×3	(1, 1)	1	-	
input	*BatchNorm*4				
Conv	3×3	(1, 1)	32	-	$\times 1$
BatchNorm	-	-	-	*lrelu*	
Conv	3×3	(1, 1)	1	-	

Table 1: Architecture of blur estimation network $\boldsymbol{B}$. The ratio of the *upsample* layers in decoder is $2\times$. We set $\alpha = 0.2$ for *lrelu* (leaky relu) activation layer.

type	size	stride	out	act.
Conv	4×4	(2, 2)	64	-
BatchNorm	-	-	-	*relu*
Conv	4×4	(2, 2)	128	-
BatchNorm	-	-	-	*relu*
Conv	4×4	(2, 2)	256	-
BatchNorm	-	-	-	*relu*
Conv	4×4	(2, 2)	512	-
BatchNorm	-	-	-	*relu*
Conv	4×4	(2, 2)	1	-

Table 2: Architecture of domain adaptation network $\boldsymbol{D}$.

type	size	stride	out	act.	repeat
Conv	1×1	(1, 1)	64	-	$\times 3$
BatchNorm	-	-	-	*relu*	
Conv	1×1	(1, 1)	32	-	$\times 3$
BatchNorm	-	-	-	*relu*	
Conv	1×1	(1, 1)	16	-	$\times 3$
BatchNorm	-	-	-	*relu*	
Conv	1×1	(1, 1)	1	-	$\times 1$

Table 3: Architecture of sharpness calibration network $\boldsymbol{S}$.

datasets	$DMENet_{BDC}$	$DMENet_{BDCS}$
SYNDOF	0.015/0.093	**0.011/0.072**
RTF	0.019/0.159	**0.012/0.088**

Table 4: Errors of the defocus maps generated with and without the sharpness calibration network on the *SYNDOF* and RTF datasets. Mean Squared Error (MSE) / Mean Absolute Error (MAE) are used as error metrics.

gives a room to $\boldsymbol{B}$ for being trained to correctly estimate a defocus map without being directly governed by L_S.

To quantitatively show the effect of sharpness calibration on the accuracy, we measured the errors of the defocus maps estimated with and without the network $\boldsymbol{S}$, using the ground truth defocus maps from the test sets of *SYNDOF* and RTF datasets. In Table 4, the errors for both synthetic and real images are reduced with sharpness calibration, which indirectly shows our framework avoids binarization degradation.

3. Quantitative Results of Ablation Study

Table 5 demonstrates the effects of incremental addition of the subnetworks to estimate defocus maps from synthetic (SYNDOF) and real (RTF and CUHK) datasets. For SYNDOF dataset, the accuracies of the results generated with the combinations vary but only in subtle degrees, as expected. However, for real images, we can easily observe positive effects of the attachments of the subnetworks, which reduces errors or increases accuracies.

	SYNDOF	RTF	CUHK
	MSE / MAE	MSE / MAE	*acc* / *mAP*
$DMENet_B$	0.010 / 0.060	0.021 / 0.113	0.732 / 0.826
$DMENet_{BD}$	**0.008 / 0.058**	0.015 / 0.095	0.797 / 0.892
$DMENet_{BC}$	0.010 / 0.067	0.021 / 0.117	0.703 / 0.812
$DMENet_{BDC}$	0.015 / 0.093	0.019 / 0.159	0.805 / 0.891
$DMENet_{BDCS}^{w/o\,L_{aux}}$	0.018 / 0.094	0.013 / 0.089	**0.880** / 0.932
$DMENet_{BDCS}$	0.011 / 0.072	**0.012 / 0.088**	0.873 / **0.935**

Table 5: Results with different combinations of subnetworks of *DMENet* in the ablation study. For SYNDOF and RTF dataset, we computed average errors in MSE/MAE. For CUHK dataset, we measured accuracy (*acc*) and mean Average Precision (*mAP*).

This confirms that the networks ***D*** and ***S*** well transfer the real domain features without harming features of synthetic images and assist ***B*** to estimate blur amounts from features of both domains.

4. Detailed Evaluation on RTF Dataset

We conducted more experiments to evaluate our network against other methods on the RTF dataset (see Table 6). We measured individual MAE for each image, as well as their average MAE/MSE. We also measured the errors on noisy versions of the images, to which we applied Gaussian noise with standard deviations of $\sigma = 1.0$ and $\sigma = 1.6$, as suggested in [1].

References

[1] L. D'Andrès, J. Salvador, A. Kochale, and S. Susstrunk. Non-parametric blur map regression for depth of field extension. *IEEE Trans. Image Processing (TIP)*, 25(4):1660–1673, 2016.

[2] A. Karaali and C. Jung. Edge-based defocus blur estimation with adaptive scale selection. *IEEE Trans. Image Processing (TIP)*, 27(3):1126–1137, 2018.

[3] J. Park, Y. Tai, D. Cho, and I. S. Kweon. A unified approach of multi-scale deep and hand-crafted features for defocus estimation. In *Proc. CVPR*, 2017.

[4] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, 2015.

[5] J. Shi, L. Xu, and J. Jia. Just noticeable defocus blur detection and estimation. In *Proc. CVPR*, 2015.

[6] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

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

Image #	[7]	[5]	[1]	[3]	[2]	Ours
01	0.229	0.146	0.098	0.106	0.119	**0.070**
02	0.358	0.303	0.129	0.135	0.172	**0.081**
03	0.233	0.150	0.106	0.133	0.186	**0.091**
04	0.216	0.226	0.080	0.099	0.114	**0.075**
05	0.211	0.157	**0.081**	0.138	0.160	0.118
06	0.210	0.155	**0.073**	0.089	0.181	0.111
07	0.230	0.158	0.105	0.122	0.185	**0.072**
08	0.490	0.253	**0.083**	0.096	0.364	0.088
09	0.404	0.273	0.069	0.132	0.224	**0.051**
10	0.268	0.189	0.131	0.180	0.128	**0.101**
11	0.400	0.187	0.112	0.086	0.190	**0.083**
12	0.432	0.325	**0.077**	0.091	0.217	0.083
13	0.258	0.275	0.084	0.134	0.147	**0.056**
14	0.343	0.303	0.266	**0.133**	0.264	0.166
15	0.535	0.248	**0.076**	0.233	0.328	0.170
16	0.289	0.249	0.108	0.124	0.221	**0.107**
17	0.485	0.365	0.134	0.156	0.289	**0.058**
18	0.324	0.135	0.105	0.111	0.181	**0.104**
19	0.319	0.416	0.135	0.130	0.226	**0.074**
20	0.329	0.141	0.112	0.094	0.158	**0.055**
21	0.296	0.245	0.094	0.106	0.116	**0.061**
22	0.437	0.402	0.084	0.219	0.202	**0.072**
Avg. MSE	0.153	0.082	0.033	0.024	0.064	**0.012**
Avg. MAE	0.332	0.241	0.106	0.129	0.199	**0.088**
Avg. Time(s)	9.42	5.30	114	12.83	1.44	**0.57**
Average results for added artificial noise ($s_\sigma = 1.0$)						
Avg. MSE	0.162	0.068	0.034	0.031	0.067	**0.020**
Avg. MAE	0.341	0.211	**0.113**	0.141	0.203	0.118
Average results for added artificial noise ($s_\sigma = 1.6$)						
Avg. MSE	0.197	0.057	0.041	**0.038**	0.073	0.046
Avg. MAE	0.370	0.184	**0.129**	0.152	0.211	0.175

Table 6: Comparison of Mean Absolute Errors (MAEs) of our *DMENet* against other competitive approaches for RTF dataset. We also show the average MAEs/MSEs for noisy versions of the dataset.