Supplementary Material of
Learning a Non-blind Deblurring Network for Night Blurry Images

Liang Chen[1*†], Jiawei Zhang[2‡†], Jinshan Pan[3], Songnan Lin[2], Faming Fang[1], Jimmy S. Ren[2,4]
[1] Shanghai Key Laboratory of Multidimensional Information Processing,
School of Computer Science and Technology, East China Normal University
[2] SenseTime Research
[3] Nanjing University of Science and Technology
[4] Qing Yuan Research Institute, Shanghai Jiao Tong University, Shanghai, China

In this supplementary material, we provide,

1. Detailed steps to solve Eq. (9) of the manuscript in Sec. 1;
2. Detailed network configurations of NBDN, including the configurations of CEU, LRU, and HPEU in Sec. 2;
3. More analyses of the proposed algorithm in Sec. 3, including:
   - Intermediate results of NBDN from different iterations in Sec. 3.1.
   - Robustness of NBDN with a larger range of noises in Sec. 3.2 and different enlarging factors in Sec. 3.3.
   - Effectiveness of the estimated confidence map from CEU in Sec. 3.4.
   - Effectiveness of the estimated hyper-parameters from HPEU in Sec. 3.5.
   - Running time comparisons in Sec. 3.6.
   - Comparison in blurry image dataset without saturation [15] in Sec. 3.7.
4. More experimental results for synthetic and real blurry images in Sec. 4.

*This work was done when Liang Chen was an intern at SenseTime.
†equal contribution
‡Corresponding author

1. Detailed steps for solving Eq. (9) in the manuscript

Eq. (9) in the manuscript is given by,

$$I^{t+1} = \arg\min_{I} \|M^t \circ (B - I \otimes K)\|^2 + \lambda \|I - U^t\|^2, \tag{1}$$

where B, I, K, M, and U are the blurry image, latent image, blur kernel, and learned prior; t is the updating index; λ is the weight hyper-parameter; we use $\otimes$ and $\circ$ to denote the convolution operation and Hadamard product. Taking the derivate of the minimization problem in Eq. (1) respect to I and setting it to zero, we have,

$$\mathbf{AI} = \mathbf{b}, \tag{2}$$

where $\mathbf{A} = (\mathbf{MK})^T\mathbf{MK} + \lambda$, $\mathbf{b} = (\mathbf{MK})^T\mathbf{MB} + \lambda\mathbf{U}$; $\mathbf{U}$, $\mathbf{I}$ and $\mathbf{B}$ denote the vectorized form of U^t, I and B, $\mathbf{K}$ is the toeplitz matrix of K w.r.t. I; $\mathbf{M}$ is M^t in a diagnoal manner.

Due to the involvement of the hardmard product operation, we cannot solve the above equation with fast Fourier transform (FFT). The overall steps for solving Eq. (1) are shown in Algorithm 1.

Algorithm 1 CG-based deconvolution step (detailed steps for solving Eq. (9) in the manuscript).

Input: $\mathbf{B}$, $\mathbf{I}$, $\mathbf{U}$, $\mathbf{M}$, $\mathbf{K}$, λ
Input: iteration number s_{max}
Output: $I_{s_{max}}$

$\mathbf{b} = \lambda\mathbf{U} + \mathbf{K}^T\mathbf{M}^T\mathbf{MB}$
$\mathbf{A} = \mathbf{K}^T\mathbf{M}^T\mathbf{MK} + \lambda$
$\mathbf{P}_0 = \mathbf{b} - \mathbf{AI}_0$
$\mathbf{r}_0 = \mathbf{P}_0$
for $s = 0$ to s_{max} **do**
  $\alpha_s = (\mathbf{r}_s^T\mathbf{r}_s)/(\mathbf{P}_s^T\mathbf{AP}_s)$
  $\mathbf{I}_{s+1} = \mathbf{I}_s + \alpha_s\mathbf{P}_s$
  $\mathbf{r}_{s+1} = \mathbf{r}_s - \alpha_s\mathbf{AP}_s$
  $\beta_s = (\mathbf{r}_{s+1}^T\mathbf{r}_{s+1})/(\mathbf{r}_s^T\mathbf{r}_s)$
  $\mathbf{P}_{s+1} = \mathbf{r}_{s+1} + \beta_s\mathbf{P}_s$
end for

$\mathbf{B}$, $\mathbf{I}$, $\mathbf{U}$, $\mathbf{M}$ are B, I, U, M in vectorized forms; $\mathbf{K}$ is the toeplitz matrix of K w.r.t. I; $\mathbf{0}$ and $\mathbf{1}$ denote the all-zero and all-one matrices.

2. Detailed network configurations of NBDN

The proposed NBDN has three individual sub-networks (i.e. CEU, LRU, and HPEU). In this section, we will give detailed configurations of each sub-network.

2.1. Configurations of CEU

CEU takes the blurry image, updated latent image, and the convolving result (i.e. B, I^t, and $I^t \otimes K$) as inputs and outputs a one-channel confidence map M^t, it is constructed with two convolution layers and three res-blocks [6], and each of the res-blocks contains two convolution layers to generate 16 features. Table 1 shows the details of CEU in one iteration. We add a rectified linear unit (ReLU) after every convolution layer except the last one for each res-block. We add the outputs from the first convolution layer and the third res-block at the end of CEU, which is further attached with a sigmoid layer to generate the final result.

2.2. Configurations of LRU

LRU takes the updated latent image I^t as input and output the prior information U^t, it is implemented by a 3-scale lightweight U-net model [12]. Specifically, each scale in the U-net model is applied with two convolutions, and each convolution layer is attached with a ReLU layer for activating. The features from the first to the last scale are 8, 16 and 32, respectively. The detailed network configurations are demonstrated in Table 2.

2.3. Configurations of HPEU

The inputs of HPEU include the updated latent image I^t and the hidden state tensor HS^t, and it generates an updated hidden state HS^{t+1} and the hyper-parameter λ^{t+1}. HPEU is constructed with eight convolution layers and a convGRU module [1] as detailed illustrated in Table 3 and Table 4. In practice, we use the first three convolution layers (i.e. conv1, conv2 and conv3 in Table 3) of HPEU to convolve I^t to obtain information x^t from the current latent image, where an adaptive pooling step is attached after the second convolution step. Then, we use the fourth convolution layer (i.e. conv4 in Table 3) of HPEU to process HS^t. The output information h^t from this step is then integrated with x^t by following steps,

$$\begin{cases} z^t = \sigma(Wz \otimes x^t + Uz \otimes h^t), & (3) \\ r^t = \sigma(Wr \otimes x^t + Ur \otimes h^t), & (4) \\ \tilde{h}^{t+1} = tanh(W \otimes x^t + U \otimes (r^t \circ h^t)), & (5) \\ HS^{t+1} = (1 - z^t) \circ h^t + z^t \circ \tilde{h}^{t+1}, & (6) \end{cases}$$

where σ denotes the sigmoid function; Wz, Uz, Wr, Ur, W and U are the kernels correspond to the layers presented in Table 4; The updated information HS^{t+1} will be used for the next updating step. To obtain the hyper-parameter, we use the last four convolution layers of HPEU (i.e. conv5 - conv8 in Table 3) to process HS^{t+1}, and then we attach a fully-connected layer at the end of HPEU to obtain the final result.

Table 1: Architecture of CEU in one iteration.

block	layer	kernel size	stride	pad	kernel number	summation
input	B, I^t, $I^t \otimes K$					
	conv1	3×3	1	1	16	
res-block1	conv2 / relu	3×3	1	1	16	
	conv3	3×3	1	1	16	conv1
res-block2	conv4 / relu	3×3	1	1	16	
	conv5	3×3	1	1	16	conv3
res-block3	conv6 / relu	3×3	1	1	16	
	conv7	3×3	1	1	16	conv1, conv5
	conv8	3×3	1	1	1	
	sigmoid					
output	M^t					

Table 2: Architecture of LRU in one iteration.

block	layer	kernel size	stride	pad	kernel number	concatenate
input	I^t					
original scale1	conv1 / relu	3×3	1	1	8	
	conv2 / relu	3×3	1	1	8	
down scale1	pool		$\downarrow 2$			
	conv3 / relu	3×3	1	1	16	
	conv4 / relu	3×3	1	1	16	
down scale2	pool		$\downarrow 2$			
	conv5 / relu	3×3	1	1	32	
	conv6 / relu	3×3	1	1	32	
down scale3	pool		$\downarrow 2$			
	conv7 / relu	3×3	1	1	32	
	conv8 / relu	3×3	1	1	32	conv6
up scale1	bilinear		$\uparrow 2$			
	conv9 / relu	3×3	1	1	16	
	conv10 / relu	3×3	1	1	16	conv4
up scale2	bilinear		$\uparrow 2$			
	conv11 / relu	3×3	1	1	8	
	conv12 / relu	3×3	1	1	8	conv2
up scale3	bilinear		$\uparrow 2$			
	conv13 / relu	3×3	1	1	8	
	conv14 / relu	3×3	1	1	8	
original scale2	conv15 / relu	3×3	1	1	1	
output	U^t					

Table 3: Architecture of HPEU

layer	kernel size	stride	pad	kernel number	input	output	output size
conv1 / relu	3×3	1	1	64	I^t		
conv2	3×3	2	1	64			
pool		adaptive					$64 \times 128 \times 128$
conv3 / relu	3×3	1	1	32		x^t	$32 \times 128 \times 128$
conv4 / relu	3×3	1	1	32	HS^t	h^t	$32 \times 128 \times 128$
convGRU					x^t, h^t	HS^{t+1}	$32 \times 128 \times 128$
conv5 / relu	3×3	2	1	32	HS^{t+1}		$32 \times 64 \times 64$
conv6 / relu	3×3	2	1	32			$32 \times 32 \times 32$
conv7 / relu	3×3	2	1	16			$16 \times 16 \times 16$
conv8	3×3	2	1	16			$16 \times 8 \times 8$
linear						λ^t	1

Table 4: Architecture of convGRU

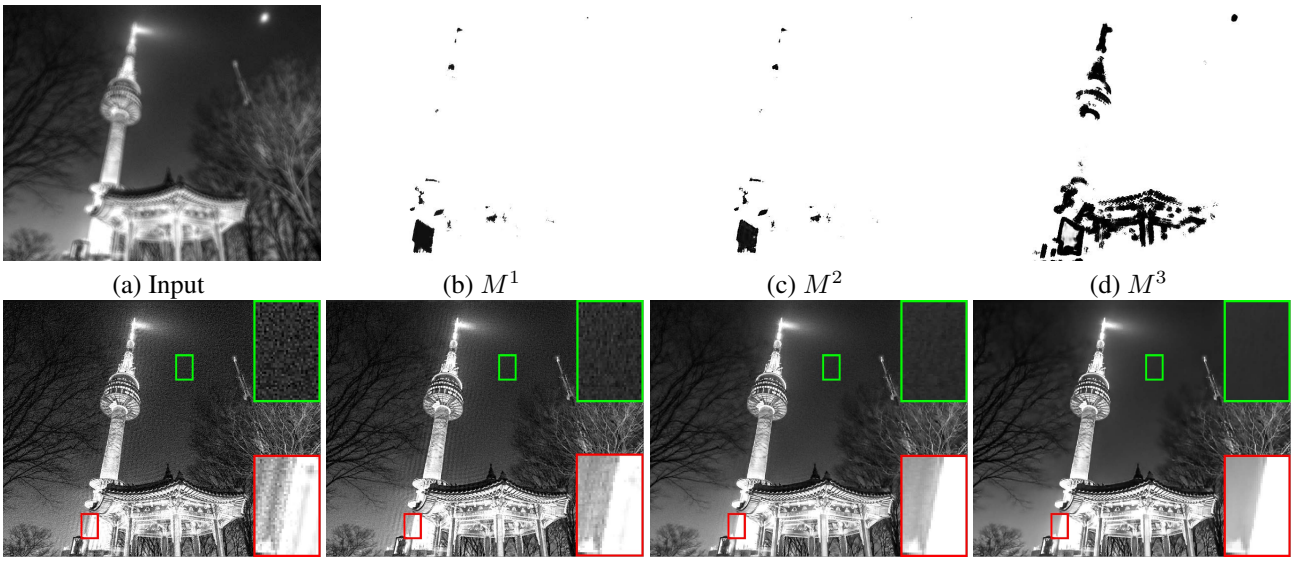
name	kernel size	stride	pad	kernel number	input	output
conv_Wz	3×3	1	1	32	x^t	
conv_Uz	3×3	1	1	32	h^t	
sigmoid						z^t
conv_Wr	3×3	1	1	32	x^t	
conv_Ur	3×3	1	1	32	h^t	
sigmoid						r^t
conv_W	3×3	1	1	32	x^t	
conv_U	3×3	1	1	32	r^t, h^t	
tanh						$\tilde{h}^{t+1}$
					z^t, h^t, $\tilde{h}^{t+1}$	HS^{t+1}

3. More Analysis

For all the compared learning-based methods, we use original implementations of IRCNN [19], RGDN [5], CPCR[4] and fine-tune the networks of FCNN [18] and SRN [16] using our training dataset. The hyper-parameters used for the optimization-based methods [2, 9, 7] are set using suggested skills from the authors for better results.

3.1. Intermediate results

Intermediate estimated latent images and confidence maps are shown in Figure 1. It shows that NBDN can progressively detect the regions that violate the linear blur model and remove noises and artifacts during the iteration process.



(a) Input (b) M^1 (c) M^2 (d) M^3

(e) Initial image I^1 (f) I^2 (g) I^3 (h) I^4

Figure 1: Intermediate results of NBDN. (b) - (d) are the intermediate confidence maps over iterations. Note M^0 is set as an all-zero matrix so we do not show it in here. (f) - (h) are the updated latent images correspond to (b) - (d). The updated latent image contains fewer noises and artifacts in the saturated regions over iterations. Please zoom-in for a better view.

3.2. Robust to noises

In our manuscript, NBDN is trained in the images with noise levels ranging from 0 to 1% [1]. In this section, we also show that NBDN can handle images with larger noises. We train NBDN with noise level ranging from 0 to 3%, and test it on the provided testing dataset with the same noise setting. Results shown in Table 5 show that NBDN still performs favorably against state-of-the-art methods when the blurry image is with large noises.

We further verify the robustness of NBDN with respect to different noises by adding only 1% or 2% random Gaussian noises to the clean images in the testing dataset. Results from different models are shown in Table 6 and Table 7. We note that NBDN can generate favorably outputs among the compared models. Even though the hyper-parameters are learned during training in FCNN [18], they are fixed in inference. With the help of the proposed HPEU, our network can dynamically adjust the hyper-parameters and handle different noise levels. That is the major reason it performs better than FCNN [18] in Table 6 and Table 7.

Table 5: Evaluations on the proposed testing dataset with 0 - 3% noises.

	Cho [2]	Hu [7]	Pan [9]	Dong [3]	SRN [16]	FCNN [18]	IRCNN [19]	RGDN [5]	CPCR [4]	Ours
Results with GT blur kernels										
PSNR	26.22	24.02	26.47	25.43	24.58	27.21	24.51	25.42	26.01	**28.06**
SSIM	0.7496	0.7144	0.7837	0.7856	0.7560	0.8500	0.6506	0.6529	0.7183	**0.8405**
Results with blur kernels from [10]										
PSNR	25.94	23.85	25.73	24.82	24.58	25.77	23.33	25.12	25.70	**27.15**
SSIM	0.7686	0.7265	0.7865	0.7780	0.7560	0.7899	0.6283	0.6595	0.7383	**0.8330**

Table 6: Average PSNR and SSIM for 1% noises on the given test dataset.

	GT kernel	kernels from [10]
Cho [2]	28.42 / 0.8716	27.11 / 0.8524
SRN [16]	24.47 / 0.7887	24.47 / 0.7887
FCNN [18]	29.03 / 0.8784	26.87 / 0.8560
RGDN [5]	27.81 / 0.8269	26.51 / 0.8293
Ours W/O HPEU	28.20 / 0.8712	26.23 / 0.8534
Ours	29.50 / 0.8845	28.50 / 0.8742

Table 7: Average PSNR and SSIM for 2% noises on the given test dataset.

	GT kernel	kernels from [10]
Cho [2]	26.54 / 0.8012	25.82 / 0.8032
SRN [16]	24.36 / 0.7848	24.36 / 0.7848
FCNN [18]	27.13 / 0.8224	26.64 / 0.8135
RGDN [5]	26.96 / 0.7378	26.49 / 0.7429
Ours W/O HPEU	26.87 / 0.8221	26.05 / 0.8154
Ours	27.86 / 0.8343	27.11 / 0.8280

[1] The 1% noise level here denotes Gaussian noise with zero mean and the standard derivation of 2.55 corresponds to the image in a range of 0 to 255.

3.3. Robust to the enlarging factors

In our manuscript, we enlarge the range of images by a factor of 1.2, and then clip the images into the dynamic range of 0 to 1 to synthesize the saturated pixels for both the training data and the test data the same as [11] (illustrated in Section 3.1 in the manuscript). To evaluate the robustness of NBDN with respect to different enlarging factors, we further train NBDN in the dataset with a dynamic range of enlarging factors (ranging from 1 to 2). The same training data is used for finetuning [18, 16]. Table 8 and Table 9 show that NBDN is also effective when the test images are synthesized by larger enlarging factors (i.e. 1.5 and 2).

Table 8: Comparison of the results from different models when the test dataset is synthesized with the enlarging factor of 1.5.

	GT kernel	kernels from [10]
Cho [2]	22.34 / 0.7467	22.02 / 0.7341
SRN [16]	22.96 / 0.7830	22.96 / 0.7830
FCNN [18]	24.41 / 0.8463	22.73 / 0.8024
RGDN [5]	23.68 / 0.7977	23.04 / 0.7838
Ours	26.17 / 0.8698	25.01 / 0.8504

Table 9: Comparison of the results from different models when the test dataset is synthesized with the enlarging factor of 2.

	GT kernel	kernels from [10]
Cho [2]	20.50 / 0.7065	20.29 / 0.6955
SRN [16]	21.83 / 0.7533	21.83 / 0.7533
FCNN [18]	21.10 / 0.7604	20.48 / 0.7298
RGDN [5]	20.76 / 0.7316	20.44 / 0.7206
Ours	23.09 / 0.8134	22.52 / 0.7963

Table 10: Comparison on the test dataset with respect to different confidence map estimation methods.

	GT blur kernels				Estimated kernels from [10]			
	M fixed as **1**	M from [2]	ad-hoc M	M from CEU	M fixed as **1**	M from [2]	ad-hoc M	M from CEU
Acc. of M	0.7929	0.7998	0.8145	0.8578	0.7854	0.7947	0.8051	0.8578
PSNR	29.36	29.47	29.45	30.06	28.20	28.25	27.92	28.45
SSIM	0.9037	0.9037	0.8930	0.9065	0.8878	0.8877	0.8775	0.8901

3.4. Effectiveness of CEU

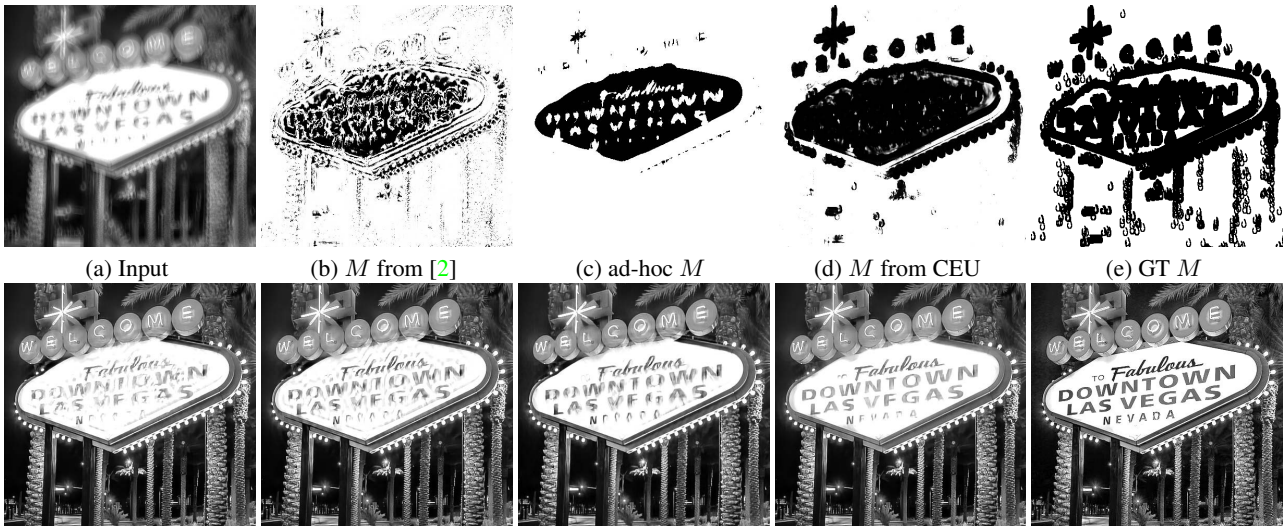
The confidence map M plays an important role in our deblurring pipeline. It detects the pixels, *i.e.* dark regions in Figure 2 (e), that do not satisfy the linear blur model:

$$B = C(K \otimes I) \neq \hat{B} = K \otimes C(I), \tag{7}$$

where $C(\cdot)$ is the clipping function that clips the saturated pixel to 1 and B is the input blurry image. $C(I)$ is the ground truth clear image and $\hat{B}$ is the blurred image generated from the ground truth clear image. We want to point out that the these regions are NOT the same with the saturated regions and most of the regions detected by Eq. (7) are the regions near the boarder of the saturated regions which are shown in Figure 2 (e). In this section, we will analysis the influence of the confidence map estimated by different methods in the proposed NBDN.

As shown in Figure 2 (f), when NBDN is trained with all the confidence values in M fixed as one, the saturated pixels will disturb the deblurring process, resulting in artifacts in the restored image. The work in [2] suggests a predefined function to compute M, which requires a heuristic guess of the density of saturated pixels, and the computed results rely heavily on the residual information (*i.e.* $B - I \otimes K$). However, Figure 2 (b) shows that this function often detects image edges instead of saturated regions, and the artifacts in the restored result can not be fully removed when we use this method to compute M while training NBDN (Figure 2 (g)). Meanwhile, we show that if M is defined in an ad-hoc strategy (pixels with values larger than a predefined threshold [2] in the blurry image are saturated, and the corresponding confidence values are set to be zeros as shown in Figure 2 (c)). However, as we analyzed before, the saturated regions in the blurry image are not the same with the pixels with low confidence. The pixels in the middle of the white areas of billboard should not have low confidence and the letters in the billboard around by saturated pixels should have low confidence. As a result, the deblurred image in Figure 2 (h) contains artifacts.

In comparison, CEU uses the blurry image and the updated latent image to guide the estimation, and it shows a better estimation result with most of the saturated pixels correspond to small confidence values. Figure 2 (i) shows that NBDN restores a visually more pleasant result when M is estimated from CEU. We further compute the accuracy of the estimated confidence maps respect to the ground truth maps. Quantitative evaluations of the four schemes on the test dataset are shown in Table 10, where M from CEU is with the highest accuracy and leads to the best performance among the compared approaches when integrated into NBDN.



(a) Input (b) M from [2] (c) ad-hoc M (d) M from CEU (e) GT M

(f) NBDN with $M = \mathbf{1}$ (g) NBDN + (b) (h) NBDN + (c) (i) NBDN + (d) (j) GT image

Figure 2: Comparisons of the results by different confidence map estimation methods. Note in (b) - (e), dark pixels indicate small confidence values. The ad-hoc strategy in (c) can be found in Section 3.4. The results demonstrate that CEU is able to generate a properer confidence map, which leads to fewer artifacts in the saturated regions.

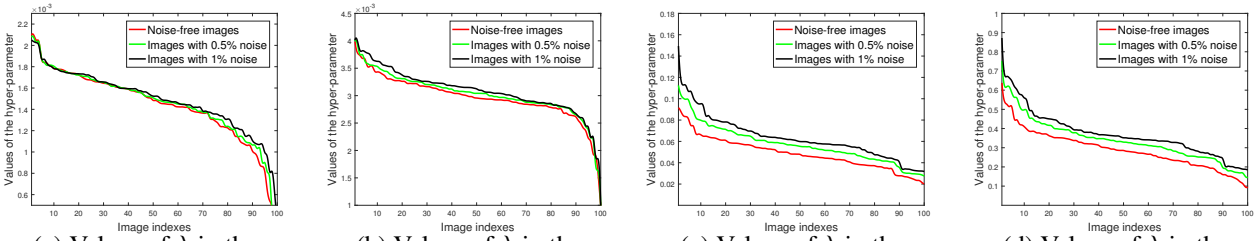
[2] We use 0.85 for the threshold value in this setting.

3.5. Effectiveness of information from the hidden state for estimation the hyper-parameters

HPEU takes advantage of both the information from the current updated latent image and the information from the previous iterations, which are stored in the hidden state (i.e. HS), to estimate the hyper-parameters. When the blurry images are contaminated with severer noises, the generated hyper-parameters are larger in most iteration stages as shown in Figure 3. Questions may be raised whether the information in HS help the current hyper-parameter estimation. To this end, we conduct an ablation study on the given test dataset to evaluate the effectiveness of the stored information in the estimation process. As shown in Table 11, HPEU can lead to better results when it is trained with information from previous iterations during the hyper-parameter estimation process.

Table 11: Effectiveness of the information from hidden state (HS) for estimating the hyper-parameters.

	GT blur kernels		Estimated kernels from [10]	
	HPEU W/O HS	HPEU with HS	HPEU W/O HS	HPEU with HS
PSNR	27.24	30.06	26.16	28.45
SSIM	0.8124	0.9065	0.7985	0.8901



(a) Values of λ in the 1st stage of NBDN (b) Values of λ in the 2nd stage of NBDN (c) Values of λ in the 3rd stage of NBDN (d) Values of λ in the 4th stage of NBDN

Figure 3: Sorted estimated hyper-parameters with respect to different levels of noise in different iteration stages of NBDN in the given test dataset.

[2]Our GPU does not have enough memory to deconvolve images with these sizes for corresponding methods.

3.6. Running time comparisons

The proposed method performs favorably against other state-of-the-art methods in terms of runtime. Table 12 summarizes the average runtime of representative methods with different image resolutions. All the methods are run on the same PC with an Intel(R) Xeon(R) CPU and an Nvidia Tesla 1080 GPU.

3.7. Images without saturation

Our method can also deblur blurry images without saturation. We compare our method on the dataset provided by Sun *et al*. [15] and compare it against state-of-the-art methods. Results in Table 13 demonstrate that our method performs favorably against existing methods when blurry images are without saturated pixels.

Table 12: Running time (seconds) comparisons on images with different sizes.

	300×300	800 × 800	1200 × 1200
Cho [2] (CPU)	5.25	77.63	169.87
Pan [9] (CPU)	15.41	239.79	574.86
FCNN [18]	0.13	N/A[3]	N/A
IRCNN [19]	1.11	2.30	5.34
RGDN [5]	5.34	23.61	N/A
Our	0.25	1.81	4.86

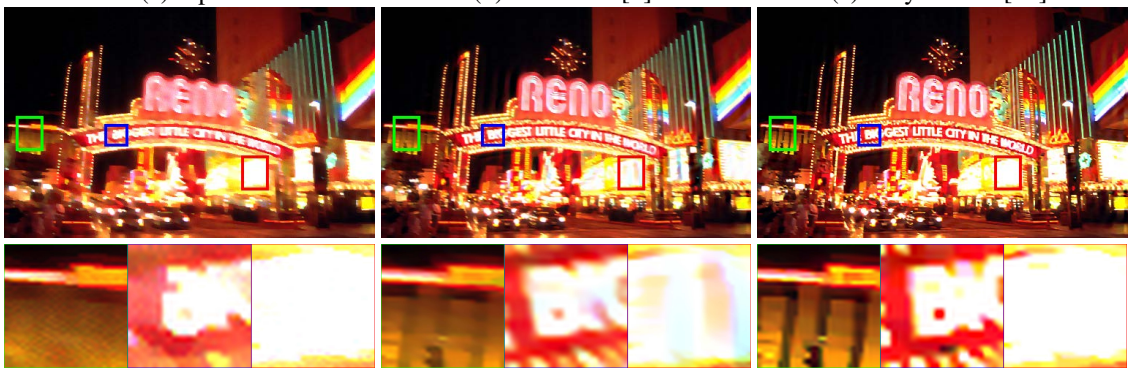
Table 13: Comparison of the results from different models on dataset without saturation [15].

	GT kernel	kernels from [9]
Krishnan [8]	31.57 / 0.87	29.94 / 0.84
Zoran [20]	33.00 / 0.89	30.61 / 0.87
Schuler [14]	31.82 / 0.86	28.76 / 0.80
Schmidt [13]	31.93 / 0.87	30.22/0.86
FCNN	32.82 / 0.90	30.39 / 0.87
Ours	32.91 / 0.90	31.54 / 0.86

4. More Examples

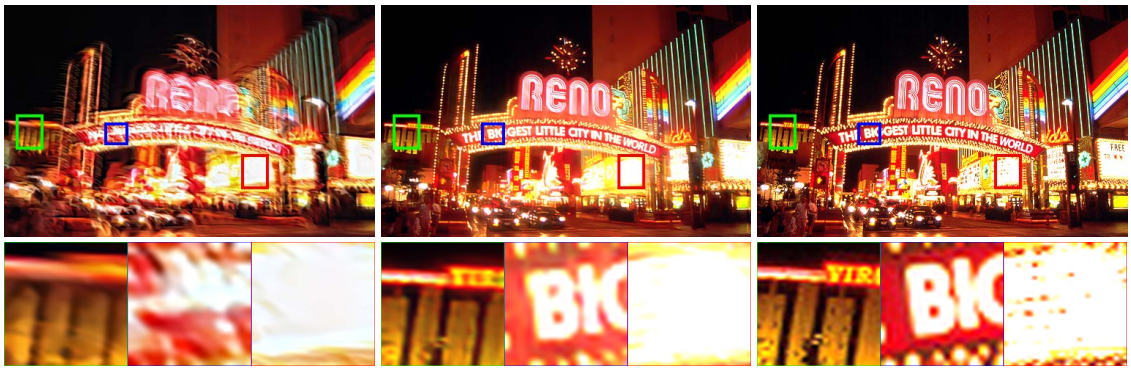
In this section, we provide more experimental examples both from the synthetic dataset and real-world. The hyper-parameters for the compared methods are tuned using the suggested skills from the authors to obtain better results.

(a) Input (b) Cho *et al*. [2] (c) Whyte *et al*. [17]



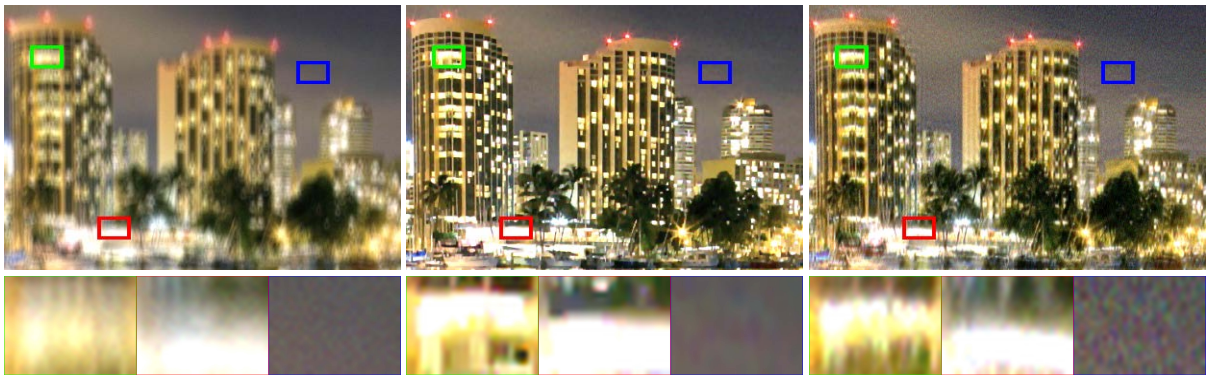
(d) Hu *et al*. [7] (e) Dong *et al*. [3] (f) Pan *et al*. [9]

(g) SRN [16] (h) FCNN [18] (i) IRCNN [19]

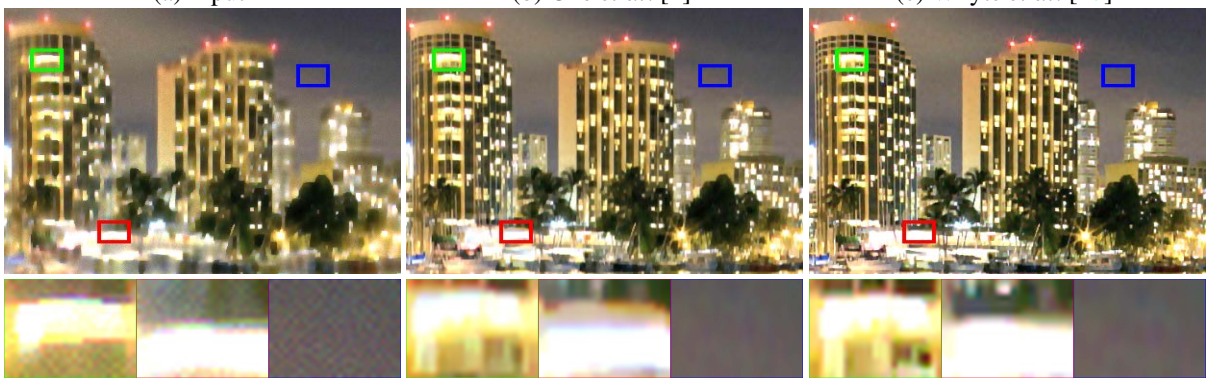


(j) RGDN [5] (k) Our (l) GT

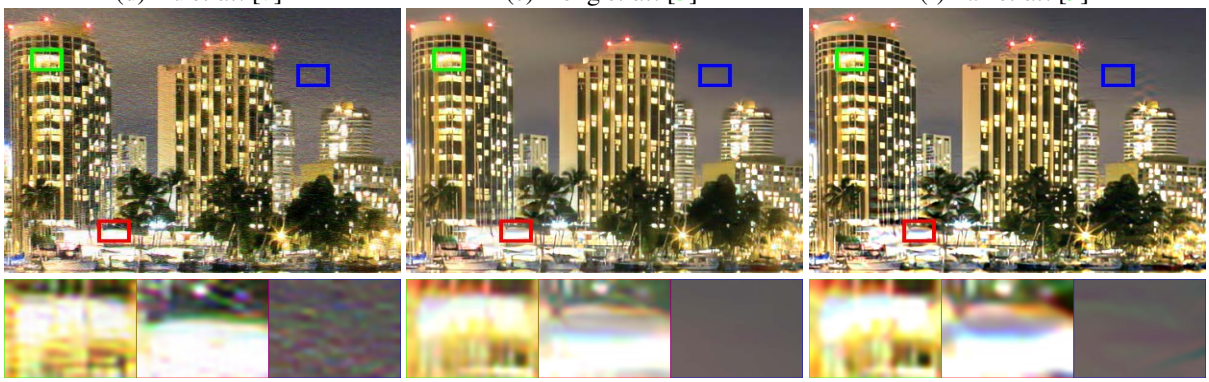
Figure 4: An example from the synthetic dataset. The robust optimization-based methods [2, 17, 7, 9] can handle the saturated pixels. But details are also erased in their results. Our method can generate a result with fine details and less artifacts in the satureated regions than other learning-based arts [18, 19, 5].



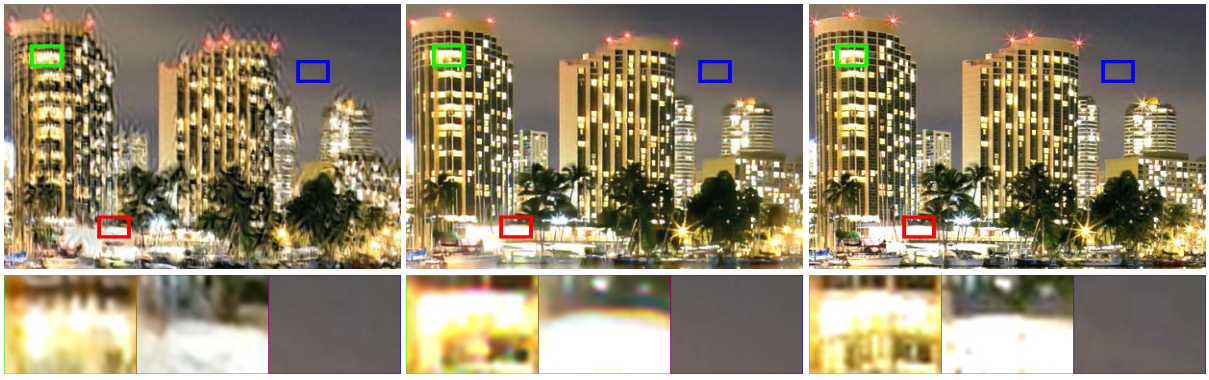
(a) Input (b) Cho *et al*. [2] (c) Whyte *et al*. [17]



(d) Hu *et al*. [7] (e) Dong *et al*. [3] (f) Pan *et al*. [9]

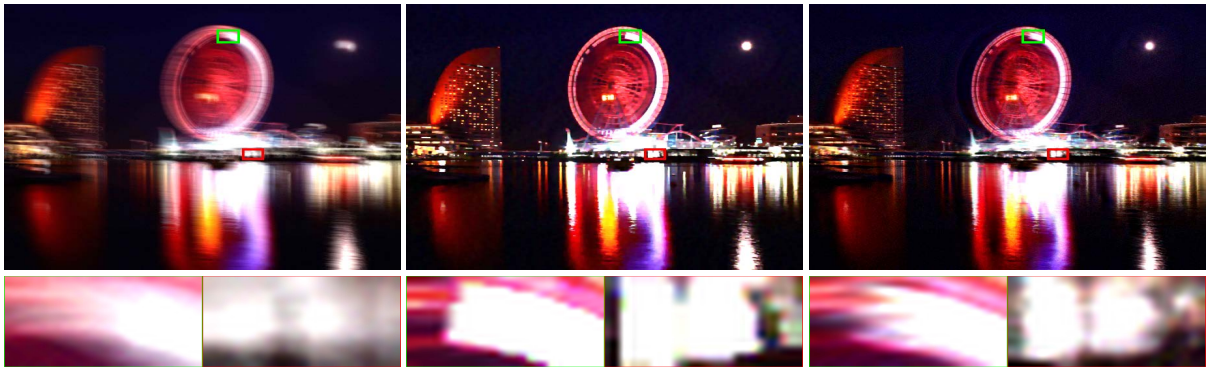


(g) SRN [16] (h) FCNN [18] (i) IRCNN [19]

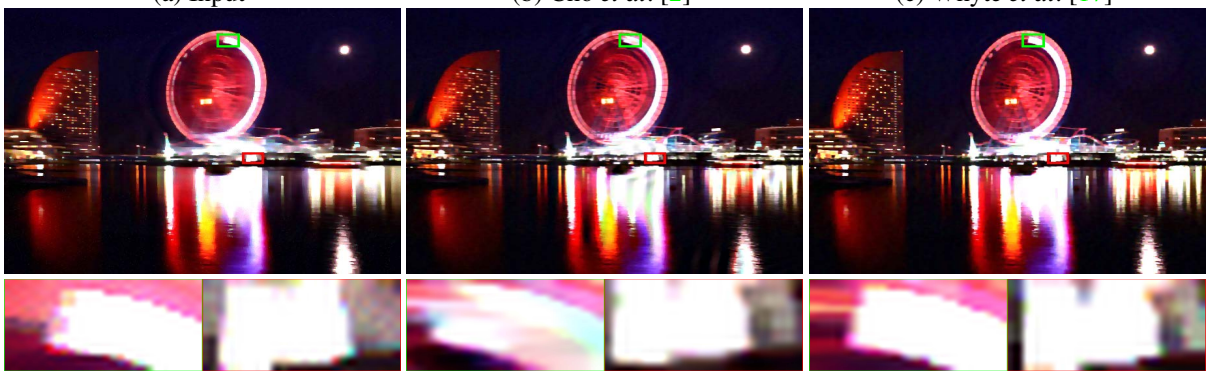


(j) RGDN [5] (k) Our (l) GT

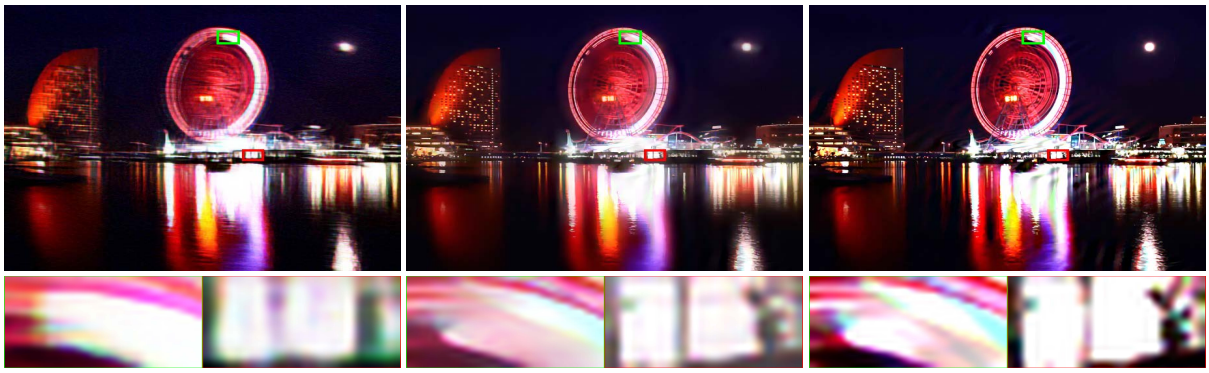
Figure 5: An example from the synthetic dataset. The robust optimization-based methods [2, 17, 7, 9] can handle the saturated pixels. But they are ineffective when the image is with severe noises. Our method can generate a result with fine details and less artifacts in the satureated regions compared to other learning-based arts [18, 19, 5].



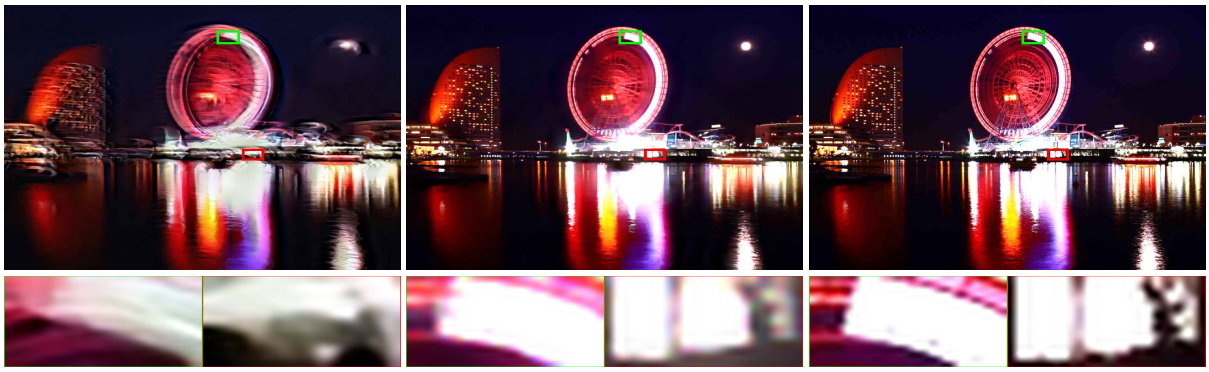
(a) Input (b) Cho *et al*. [2] (c) Whyte *et al*. [17]



(d) Hu *et al*. [7] (e) Dong *et al*. [3] (f) Pan *et al*. [9]

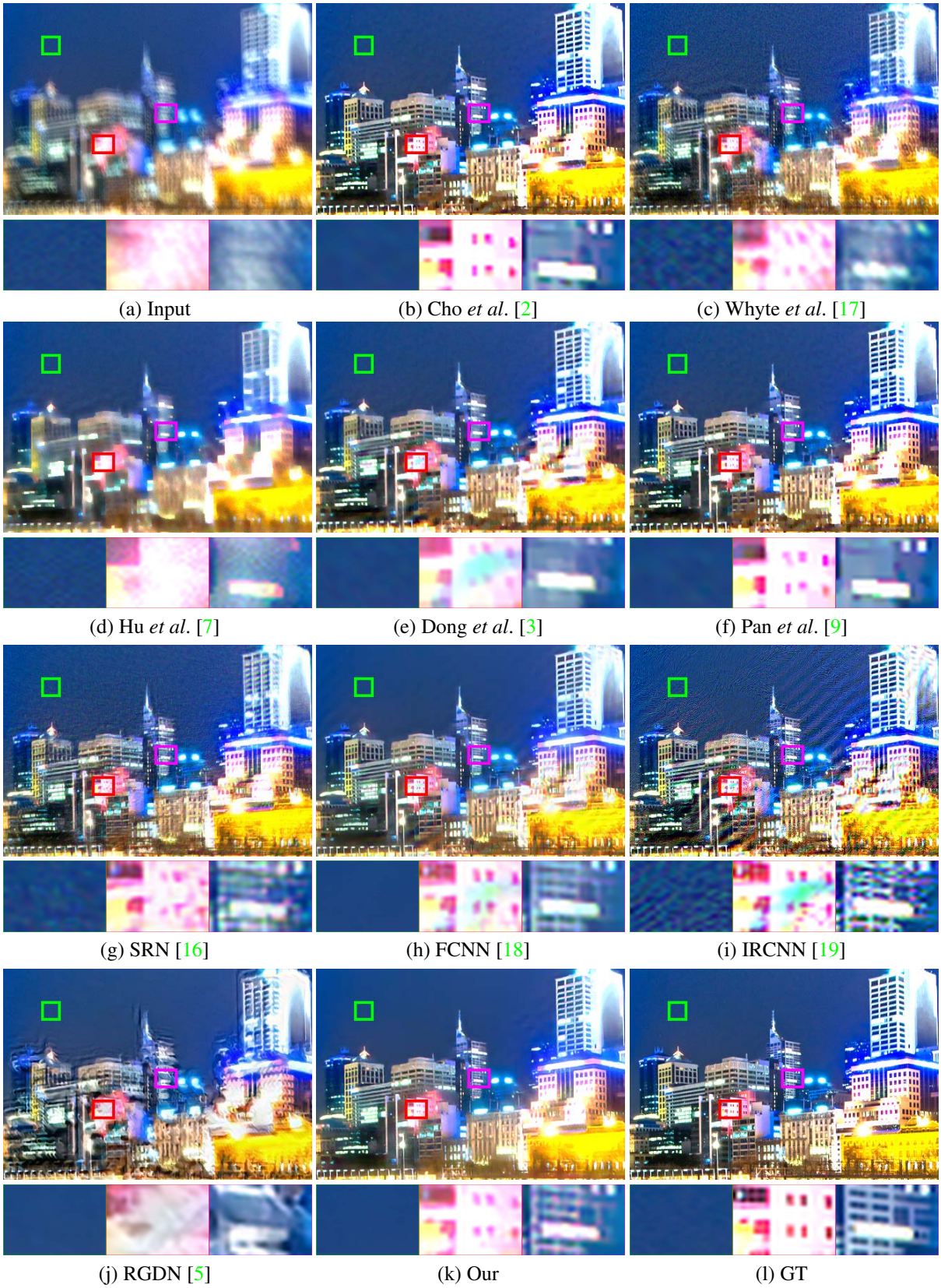


(g) SRN [16] (h) FCNN [18] (i) IRCNN [19]



(j) RGDN [5] (k) Our (l) GT

Figure 6: An example from the synthetic dataset. The robust optimization-based methods [2, 17, 7, 9] can handle the saturated pixels. But details are erased in their results. Our method can generate a result with fine details and less artifacts in the satureated regions compared to other learning-based arts [18, 19, 5].



(a) Input (b) Cho *et al*. [2] (c) Whyte *et al*. [17]

(d) Hu *et al*. [7] (e) Dong *et al*. [3] (f) Pan *et al*. [9]

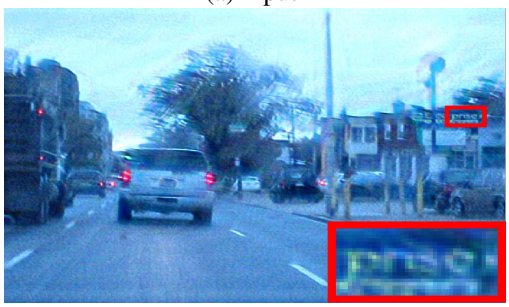
(g) SRN [16] (h) FCNN [18] (i) IRCNN [19]

(j) RGDN [5] (k) Our (l) GT

Figure 7: An example from the synthetic dataset. The robust optimization-based methods [2, 17, 7, 9] can handle the saturated pixels. But they are ineffective when the image is with severe noises. Our method can generate a result with fine details and less artifacts in the satureated regions compared to other learning-based arts [18, 19, 5].

(a) Input

(b) Cho *et al*. [2]



(c) Whyte *et al*. [17]

(d) Hu *et al*. [7]

(e) Dong *et al*. [3]

(f) Pan *et al*. [9]

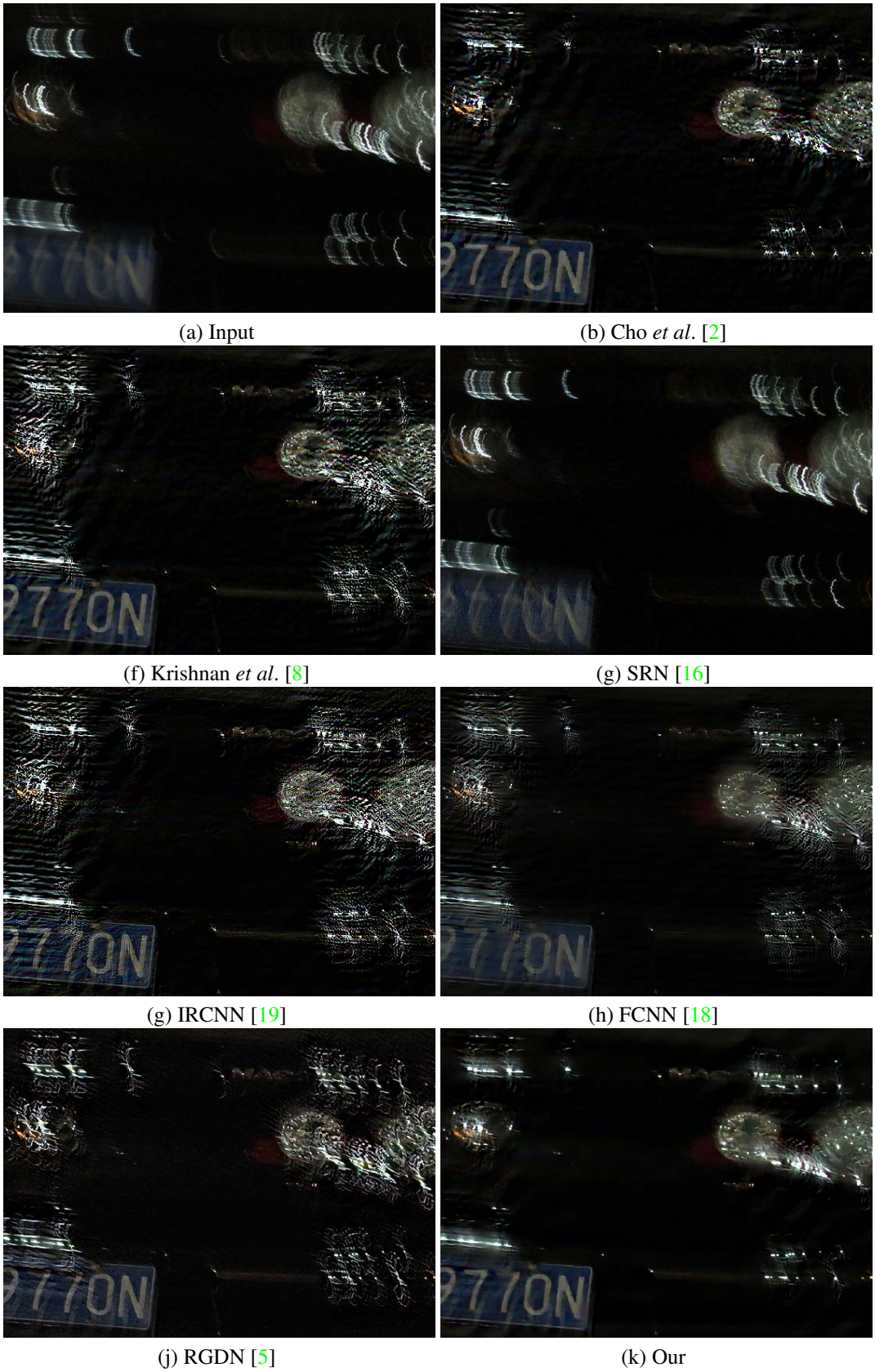
(g) SRN [16]

(h) FCNN [18]

(j) RGDN [5]

(k) Our

Figure 8: A real example with saturated pixels. Our method generates comparable results with state-of-the-art methods.



(a) Input (b) Cho *et al*. [2]

(f) Krishnan *et al*. [8] (g) SRN [16]

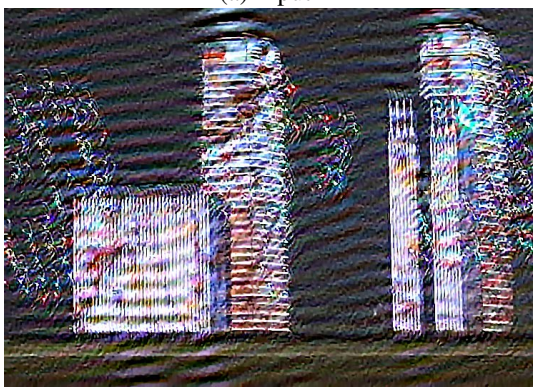
(g) IRCNN [19] (h) FCNN [18]

(j) RGDN [5] (k) Our

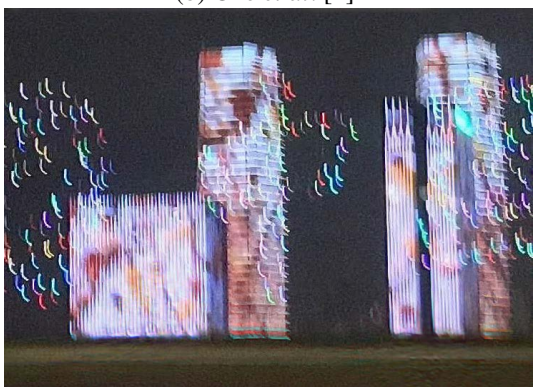
Figure 9: A real example in a severely low-light condition. Our method generates comparable results with other models.

(a) Input

(b) Cho *et al*. [2]



(f) Krishnan *et al*. [8]



(g) SRN [16]

(g) IRCNN [19]

(h) FCNN [18]

(j) RGDN [5]

(k) Our

Figure 10: A real example in a severely low-light condition. Our method generates comparable results with other models.

References

[1] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015. 3

[2] Sunghyun Cho, Jue Wang, and Seungyong Lee. Handling outliers in non-blind image deconvolution. In *ICCV*, 2011. 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17

[3] Jiangxin Dong, Jinshan Pan, Deqing Sun, Zhixun Su, and Ming-Hsuan Yang. Learning data terms for non-blind deblurring. In *ECCV*, 2018. 6, 11, 12, 13, 14, 15

[4] Thomas Eboli, Jian Sun, and Jean Ponce. End-to-end interpretable learning of non-blind image deblurring. *arXiv preprint arXiv:2007.01769*, 2020. 5, 6

[5] Dong Gong, Zhen Zhang, Qinfeng Shi, Anton van den Hengel, Chunhua Shen, and Yanning Zhang. Learning an optimizer for image deconvolution. *IEEE TNNLS*, 2020. 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[7] Zhe Hu, Sunghyun Cho, Jue Wang, and Ming-Hsuan Yang. Deblurring low-light images with light streaks. In *CVPR*, 2014. 5, 6, 11, 12, 13, 14, 15

[8] Dilip Krishnan and Rob Fergus. Fast image deconvolution using hyper-laplacian priors. In *NIPS*, 2009. 10, 16, 17

[9] Jinshan Pan, Zhouchen Lin, Zhixun Su, and Ming-Hsuan Yang. Robust kernel estimation with outliers handling for image deblurring. In *CVPR*, 2016. 5, 6, 10, 11, 12, 13, 14, 15

[10] Jinshan Pan, Deqing Sun, Hanspeter Pfister, and Ming-Hsuan Yang. Blind image deblurring using dark channel prior. In *CVPR*, 2016. 6, 7, 9

[11] Wenqi Ren, Jiawei Zhang, Lin Ma, Jinshan Pan, Xiaochun Cao, Wangmeng Zuo, Wei Liu, and Ming-Hsuan Yang. Deep non-blind deconvolution via generalized low-rank approximation. In *NIPS*, 2018. 7

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 3

[13] Uwe Schmidt and Stefan Roth. Shrinkage fields for effective image restoration. In *CVPR*, 2014. 10

[14] Christian J Schuler, Harold Christopher Burger, Stefan Harmeling, and Bernhard Scholkopf. A machine learning approach for non-blind image deconvolution. In *CVPR*, 2013. 10

[15] Libin Sun, Sunghyun Cho, Jue Wang, and James Hays. Edge-based blur kernel estimation using patch priors. In *IEEE International Conference on Computational Photography*, 2013. 1, 10

[16] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *CVPR*, 2018. 5, 6, 7, 11, 12, 13, 14, 15, 16, 17

[17] Oliver Whyte, Josef Sivic, and Andrew Zisserman. Deblurring shaken and partially saturated images. *IJCV*, 2014. 11, 12, 13, 14, 15

[18] Jiawei Zhang, Jinshan Pan, Wei-Sheng Lai, Rynson W. H. Lau, and Ming-Hsuan Yang. Learning fully convolutional networks for iterative non-blind deconvolution. In *CVPR*, 2017. 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17

[19] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep cnn denoiser prior for image restoration. In *CVPR*, 2017. 5, 6, 10, 11, 12, 13, 14, 16, 17

[20] Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. In *ICCV*, 2011. 10