"A Model-driven Deep Neural Network for Single Image Rain Removal": Supplementary Material

Hong Wang[1], Qi Xie[1], Qian Zhao[1], Deyu Meng[2,1]
[1]Xi'an Jiaotong University; [2]Macau University of Science and Technology
{hongwang01,xq.liwu}@stu.xjtu.edu.cn timmy.zhaoqian@gmail.com dymeng@mail.xjtu.edu.cn

Abstract

In this supplementary material, we provide more details on the optimization algorithm, network design, and parameter settings in our experiments. Besides, we show more analysis on our model and provide more ablation studies about the proposed rain convolutional dictionary network (RCDNet). In the end, we utilize more representative rainy images with various rain patterns to demonstrate more experimental results for rain removal performance comparisons and model verification.

1. More details of optimization algorithm

In this section, we provide a detailed derivation for optimization algorithm in Section 3.2 of the main text. The expression of the original optimization problem is:

$$\min_{\mathcal{M},\mathcal{B}} \left\| \mathcal{O}-\mathcal{B}-\sum_{n=1}^{N} \mathcal{C}_n \otimes \boldsymbol{M}_n \right\|_F^2 + \alpha g_1(\mathcal{M}) + \beta g_2(\mathcal{B}), \quad (1)$$

where $\mathcal{M} \in \mathbb{R}^{H\times W\times N}$ is the tensor form of $\boldsymbol{M}_n$s. α and β are trade-off parameters. $g_1(\cdot)$ and $g_2(\cdot)$ represent the regularizers to deliver the prior structures of $\boldsymbol{M}_n$ and $\mathcal{B}$, respectively.

As explained in the main text, we prefer to build a new algorithm for solving the problem through alternately updating $\mathcal{M}$ and $\mathcal{B}$ by the proximal gradient technique [1]. The details are as follows:

Updating $\mathcal{M}$: The rain maps $\mathcal{M}$ can be updated by solving:

$$\mathcal{M}^{(s)} = \arg\min_{\mathcal{M}} Q_1\left(\mathcal{M}, \mathcal{M}^{(s-1)}\right), \quad (2)$$

where $\mathcal{M}^{(s-1)}$ is the updating result obtained in the last iteration, and $Q_1(\mathcal{M}, \mathcal{M}^{(s-1)})$ is a quadratic approximation of the objective function (1) with respect to $\mathcal{M}$ [1], expressed as:

$$\begin{aligned} Q_1\left(\mathcal{M}, \mathcal{M}^{(s-1)}\right) = & f\left(\mathcal{M}^{(s-1)}\right) + \frac{1}{2\eta_1}\left\|\mathcal{M}-\mathcal{M}^{(s-1)}\right\|_F^2 \\ & + \left\langle \mathcal{M}-\mathcal{M}^{(s-1)}, \nabla f\left(\mathcal{M}^{(s-1)}\right)\right\rangle + \alpha g_1(\mathcal{M}), \end{aligned} \quad (3)$$

where $f(\mathcal{M}^{(s-1)}) = \left\|\mathcal{O}-\mathcal{B}^{(s-1)}-\sum_{n=1}^{N}\mathcal{C}_n\otimes\boldsymbol{M}_n^{(s-1)}\right\|_F^2$ and η_1 is the stepsize parameter. It is easy to prove that the problem (2) is equivalent to:

$$\min_{\mathcal{M}} \frac{1}{2}\left\|\mathcal{M}-\left(\mathcal{M}^{(s-1)}-\eta_1\nabla f\left(\mathcal{M}^{(s-1)}\right)\right)\right\|_F^2 + \alpha\eta_1 g_1(\mathcal{M}). \quad (4)$$

Corresponding to general regularization terms [2], the solution of Eq. (4) is:

$$\mathcal{M}^{(s)} = \text{prox}_{\alpha\eta_1}\left(\mathcal{M}^{(s-1)}-\eta_1\nabla f\left(\mathcal{M}^{(s-1)}\right)\right). \quad (5)$$

Moreover, by substituting

$$\nabla f\left(\mathcal{M}^{(s-1)}\right) = \mathcal{C}\otimes^T\left(\sum_{n=1}^{N}\mathcal{C}_n\otimes\boldsymbol{M}_n^{(s-1)}+\mathcal{B}^{(s-1)}-\mathcal{O}\right), \quad (6)$$

where $\mathcal{C}\in\mathbb{R}^{k\times k\times N\times 3}$ is a 4-D tensor stacked by $\mathcal{C}_n$s, and $\otimes^T$ denotes the transposed convolution[1], we can obtain the updating formula for $\mathcal{M}$ as[2]:

$$\mathcal{M}^{(s)} = \text{prox}_{\alpha\eta_1}\left(\mathcal{M}^{(s-1)}-\eta_1\mathcal{C}\otimes^T\left(\sum_{n=1}^{N}\mathcal{C}_n\otimes\boldsymbol{M}_n^{(s-1)}+\mathcal{B}^{(s-1)}-\mathcal{O}\right)\right), \quad (7)$$

where $\text{prox}_{\alpha\eta_1}(\cdot)$ is the proximal operator dependent on the regularization term $g_1(\cdot)$ with respect to $\mathcal{M}$.

Updating $\mathcal{B}$: Similarly, the quadratic approximation of the objective function (1) with respect to $\mathcal{B}$ is:

$$\begin{aligned} Q_2\left(\mathcal{B}, \mathcal{B}^{(s-1)}\right) = & h\left(\mathcal{B}^{(s-1)}\right) + \frac{1}{2\eta_2}\left\|\mathcal{B}-\mathcal{B}^{(s-1)}\right\|_F^2 \\ & + \left\langle \mathcal{B}-\mathcal{B}^{(s-1)}, \nabla h\left(\mathcal{B}^{(s-1)}\right)\right\rangle + \beta g_2(\mathcal{B}), \end{aligned} \quad (8)$$

where $h(\mathcal{B}^{(s-1)}) = \left\|\mathcal{O}-\mathcal{B}^{(s-1)}-\sum_{n=1}^{N}\mathcal{C}_n\otimes\boldsymbol{M}_n^{(s)}\right\|_F^2$ and η_2 is the stepsize parameter. Then the equivalent optimization problem is:

$$\min_{\mathcal{B}} \frac{1}{2}\left\|\mathcal{B}-\left(\mathcal{B}^{(s-1)}-\eta_2\nabla h\left(\mathcal{B}^{(s-1)}\right)\right)\right\|_F^2 + \beta\eta_2 g_2(\mathcal{B}). \quad (9)$$

[1]For any tensor $\mathcal{A}\in\mathbb{R}^{H\times W\times 3}$, we can calculate the n^{th} channel of $\mathcal{C}\otimes^T\mathcal{A}$ by $\sum_{c=1}^{3}\mathcal{C}_{\{:,:,n,c\}}\otimes^T\mathcal{A}_{\{:,:,c\}}$.

[2]It can be proved that, with small enough η_1 and η_2, Eq. (7) and Eq. (10) can both lead to the reduction of objective function (1) [1].

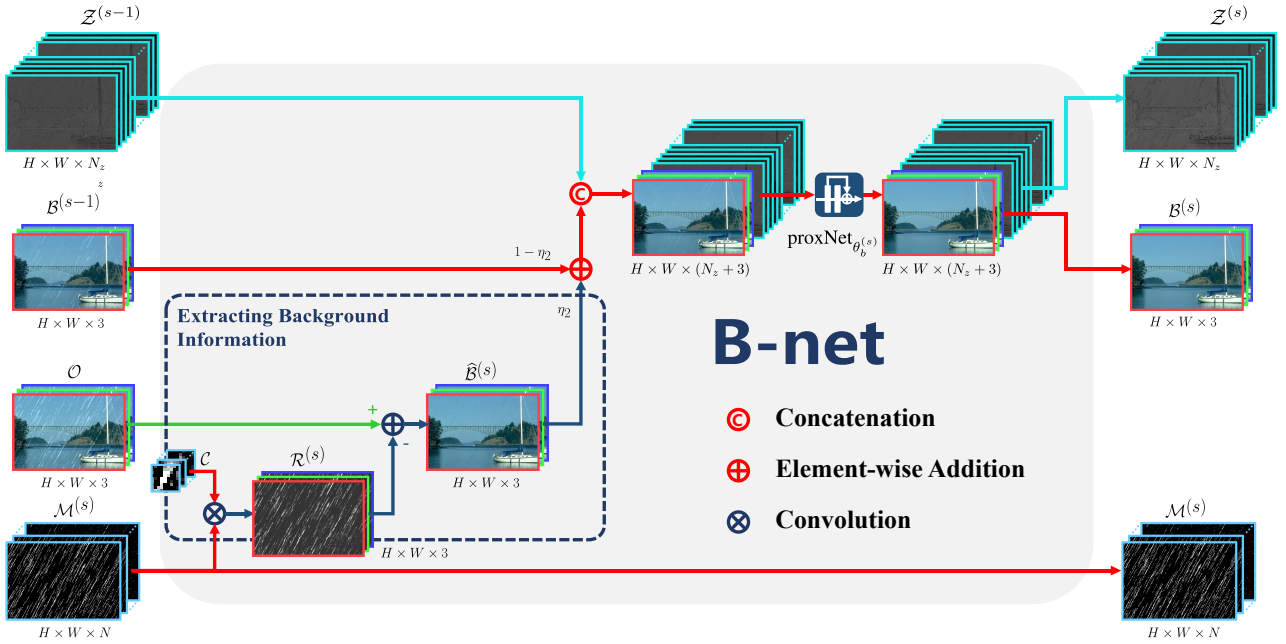

Figure 1. Illustration of B-net with channel concatenation operator at the s^{th} stage. The images are better observed by zooming in on screen.

With $\nabla h\left(\mathcal{B}^{(s-1)}\right) = \sum_{n=1}^{N} \mathcal{C}_n \otimes \boldsymbol{M}_n^{(s)} + \mathcal{B}^{(s-1)} - \mathcal{O}$, it is easy to deduce that the final updating rule for $\mathcal{B}$ is[2]:

$$\mathcal{B}^{(s)} = \text{prox}_{\beta\eta_2}\left((1-\eta_2)\,\mathcal{B}^{(s-1)} + \eta_2\left(\mathcal{O} - \sum_{n=1}^{N} \mathcal{C}_n \otimes \boldsymbol{M}_n^{(s)}\right)\right), \tag{10}$$

where $\text{prox}_{\beta\eta_2}(\cdot)$ is the proximal operator correlated to the regularization term $g_2(\cdot)$ with respect to $\mathcal{B}$.

Here, Eq. (4) and Eq. (9) correspond to Eq. (5) and Eq. (9) in the main text, respectively.

2. More details of network design

In this section, we present more details of the network design, including the channel concatenation operator, initialization at $s = 0$, $\text{proxNet}_{\theta_m^{(s)}}(\cdot)$, and $\text{proxNet}_{\theta_b^{(s)}}(\cdot)$.

Channel concatenation operator. As stated in Remark presented in Section 4.1 of the main text, the input tensor of $\text{proxNet}_{\theta_b^{(s)}}(\cdot)$ is $\left((1-\eta_2)\mathcal{B}^{(s-1)} + \eta_2\widehat{\mathcal{B}}^{(s)}\right)$ which has the same size of $H \times W \times 3$ as the to-be-estimated $\mathcal{B}$. Evidently, this is not beneficial for learning $\mathcal{B}$ since most of the previous updating information would be compressed due to not sufficiently specified number of channels. To better keep and deliver image features, we simply expand the input tensor at the 3rd mode for more channels in experiments.

Specifically, we introduce an auxiliary variable $\mathcal{Z}$ with the size of $H \times W \times N_z$, put it behind the tensor $\left((1-\eta_2)\mathcal{B}^{(s-1)} + \eta_2\widehat{\mathcal{B}}^{(s)}\right)$ at the channel mode, and then obtain a new input tensor with the size of $H \times W \times (N_z + 3)$ as the input of $\text{proxNet}_{\theta_b^{(s)}}(\cdot)$. Thus the number of input channels of $\text{proxNet}_{\theta_b^{(s)}}(\cdot)$ varies from 3 to $(N_z + 3)$. Accordingly, for the output with the size $H \times W \times (N_z + 3)$ of $\text{proxNet}_{\theta_b^{(s)}}(\cdot)$, we decompose it into two sub-tensors based on the channel mode: one sub-tensor corresponding to the first 3 channels is taken as the updated background layer $\mathcal{B}^{(s)}$ and the other one corresponding to remaining channels is regraded as $\mathcal{Z}^{(s)}$ with the size of $H \times W \times N_z$. Please refer to Fig. 1 for better understanding.

Initialize $\mathcal{B}^{(0)}$ and $\mathcal{Z}^{(0)}$. In our network implementation, we set $\mathcal{M}^{(0)} = 0$, and initialize $\mathcal{B}^{(0)}$ and $\mathcal{Z}^{(0)}$ by:

$$\{\mathcal{B}^{(0)} \mid \mathcal{Z}^{(0)}\} = \text{proxNet}_{\theta_b^{(0)}}\left(\text{concat}\left(\mathcal{O}, \mathcal{C}_z \otimes \mathcal{O}\right)\right), \tag{11}$$

where $\mathcal{C}_z$ is the learnable convolutional filters with the size of $k_z \times k_z \times 3 \times N_z$. Refer to Eq. (2) in the main text for better understanding of the convolutional operator. Actually, this has been ready-made in current popular deep learning (DL) framework such as Tensorflow[3] and PyTorch[4]. The function concat$(\cdot)$ means the channel concatenation operator as illustrated before. Here the operator is executed between $\mathcal{O}$ and $\mathcal{C}_z \otimes \mathcal{O}$. $\text{proxNet}_{\theta_b^{(0)}}(\cdot)$ is a deep residual network (ResNet) [6] with the same structure as $\text{proxNet}_{\theta_b^{(s)}}(\cdot)$ $(s = 1, \cdots, S)$, but with different parameter as $\theta_b^{(0)}$. Clearly, the output tensor of $\text{proxNet}_{\theta_b^{(0)}}(\cdot)$ is also

[3] https://tensorflow.google.cn/
[4] https://pytorch.org/docs/stable/index.html

Table 1. Rain removal performance on Rain100L of the proposed RCDNet with different loss functions. Note that the tradeoff parameter not mentioned in each version is 0 by default. For example, for Version 1, $\lambda_s = 0$ $(s = 0, \cdots, S-1)$ and $\gamma_s = 0$ $(s = 1, \cdots, S)$.

Version	Parameter setting	Loss function	PSNR	SSIM
1	$\lambda_S = 1$	$L = \left\|\mathcal{B}^{(S)} - \mathcal{B}\right\|_F^2$	39.90	0.9855
2	$\lambda_S = 1$ $\lambda_s = 0.1 (s \neq S)$	$L = \left\|\mathcal{B}^{(S)} - \mathcal{B}\right\|_F^2 + \sum_{s=0}^{S-1} 0.1 \left\|\mathcal{B}^{(s)} - \mathcal{B}\right\|_F^2$	39.93	0.9857
3	$\lambda_S = \gamma_S = 1$	$L = \left\|\mathcal{B}^{(S)} - \mathcal{B}\right\|_F^2 + \left\|\mathcal{O} - \mathcal{B} - \mathcal{R}^{(S)}\right\|_F^2$	39.94	0.9860
4	$\lambda_S = \gamma_S = 1$ $\gamma_s = 0.1 (s \neq S)$	$L = \left\|\mathcal{B}^{(S)} - \mathcal{B}\right\|_F^2 + \left\|\mathcal{O} - \mathcal{B} - \mathcal{R}^{(S)}\right\|_F^2 + \sum_{s=1}^{S-1} 0.1 \left\|\mathcal{O} - \mathcal{B} - \mathcal{R}^{(s)}\right\|_F^2$	39.98	0.9860
5	$\lambda_S = \gamma_S = 1$ $\lambda_s = \gamma_s = 0.1 (s \neq S)$	$L = \left\|\mathcal{B}^{(S)} - \mathcal{B}\right\|_F^2 + \left\|\mathcal{O} - \mathcal{B} - \mathcal{R}^{(S)}\right\|_F^2 + \sum_{s=0}^{S-1} 0.1 \left\|\mathcal{B}^{(s)} - \mathcal{B}\right\|_F^2 + \sum_{s=1}^{S-1} 0.1 \left\|\mathcal{O} - \mathcal{B} - \mathcal{R}^{(s)}\right\|_F^2$	**40.00**	**0.9860**

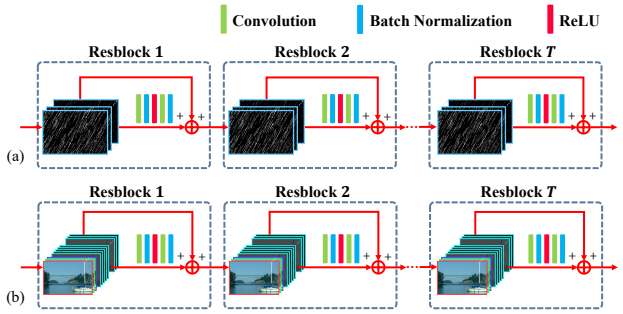

Figure 2. (a) The exploited ResNet for the proximal network $\text{proxNet}_{\theta_m^{(s)}}(\cdot)$. (b) The exploited ResNet for the proximal network $\text{proxNet}_{\theta_b^{(s)}}(\cdot)$.

with the size of $H \times W \times (N_z + 3)$, and can be decomposed into $\mathcal{B}^{(0)}$ with the size of $H \times W \times 3$ and $\mathcal{Z}^{(0)}$ with the size of $H \times W \times N_z$ based on the channel mode, as depicted above. In this manner, we keep the number of input and output channels through $\text{proxNet}_{\theta_b^{(0)}}(\cdot)$ consistent, and take the corresponding fine-tuned results of $\mathcal{O}$ and $\mathcal{C}_z \otimes \mathcal{O}$ with the ResNet as $\mathcal{B}^{(0)}$ and $\mathcal{Z}^{(0)}$, respectively. Intuitively, this is simple and reasonable.

Proximal Network and ResNet. As stated before, we adopt the ResNet to build the proximal network $\text{proxNet}_{\theta_m^{(s)}}(\cdot)$ and $\text{proxNet}_{\theta_b^{(s)}}(\cdot)$ $(s = 0, 1, \cdots, S)$. Refer to Fig. 2 (a) and Fig. 2 (b) for detailed illustration of the used ResNet for $\text{proxNet}_{\theta_m^{(s)}}(\cdot)$ and $\text{proxNet}_{\theta_b^{(s)}}(\cdot)$, respectively. For simplicity, in all experiments, among all stages, the number T of Resblocks in $\text{proxNet}_{\theta_m^{(s)}}(\cdot)$ and $\text{proxNet}_{\theta_b^{(s)}}(\cdot)$ is with the same setting.

Here, all the parameters involved in the RCDNet can be automatically learned from training data in an end-to-end manner, including $\{\theta_m^{(s)}, \theta_b^{(s)}\}_{s=1}^S$, rain kernels $\mathcal{C}$, η_1, and η_2, $\mathcal{C}_z$, and $\theta_b^{(0)}$.

3. More implementation details

We use PyTorch [11] to implement our network, based on a PC equipped with an Intel (R) Core(TM) i7-8700K at 3.70GHZ and a NVIDIA GeForce GTX 1080Ti GPU. We adopt the Adam optimizer [8] with the batch size of 16 and the patch size of 64×64. The initial learning rate is 1×10^{-3} and divided by 5 every 25 epochs. The total epoch is 100. It is worth mentioning that for all datasets in the main text, these parameter settings are the same. This would show the favorable robustness and generality of our method.

4. More ablation studies

In this section, we provide more ablation studies based on Rain100L. The synthesized dataset consists of 200 rainy/clean image pairs for training and 100 pairs for testing [16]. Two performance metrics are employed, including peak-signal-to-noise ratio (PSNR) [7] and structure similarity (SSIM) [14]. As the human visual system is sensitive to the Y channel of a color image in YCbCr space, we compute PSNR and SSIM based on this luminance channel.

4.1. Loss function

As stated in the main text, the objective function for training the proposed RCDNet is expressed as:

$$L = \sum_{s=0}^{S} \lambda_s \left\|\mathcal{B}^{(s)} - \mathcal{B}\right\|_F^2 + \sum_{s=1}^{S} \gamma_s \left\|\mathcal{O} - \mathcal{B} - \mathcal{R}^{(s)}\right\|_F^2, \quad (12)$$

where $\mathcal{B}^{(s)}$ and $\mathcal{R}^{(s)}$ denote the derained result and extracted rain layer at the s^{th} stage, respectively. λ_s and γ_s are tradeoff parameters.

Here, we choose the number N of rain maps and the channel concatenation number N_z as 32. The size k of rain kernels is 9, and $k_z = 3$. Setting the number T of Resblocks in each ResNet at every stage as 4 and the stage number S as 17, we study the effect of loss function with different parameter settings of λ_s and γ_s on rain removal performance of RCDNet, as depicted in Table 1. From Version 1, we can find that even when we only adopt a single loss on the final derained result $\widehat{\mathcal{B}}$, our method can still significantly outperform other comparison methods on Rain100L, as reported in Table 5. By comparing Versions 1 and 3, we can see that the performance can be further improved by imposing supervision loss on the final extracted rain layer $\mathcal{R}^{(S)}$. Besides, intra-stage supervision for the results $\mathcal{B}^{(s)}\left(\mathcal{R}^{(s)}\right)$ is helpful for enabling the network to be evolved to a better direction. This can be easily concluded by comparing

Table 2. Effect of Resblocks number T, involved in the ResNet as shown in Fig. 2, on the performance of RCDNet.

T	T=1	T=2	T=3	T=4	T=5
PSNR	39.04	39.52	39.80	40.00	39.98
SSIM	0.9833	0.9848	0.9856	0.9860	0.9859

Table 3. Effect of stage number S on the performance of RCDNet.

Stage No.	S=0	S=2	S=5	S=8	S=11	S=14	S=17	S=20
PSNR	35.93	38.46	39.35	39.60	39.81	39.90	40.00	39.91
SSIM	0.9689	0.9813	0.9842	0.9850	0.9855	0.9858	0.9860	0.9858

Table 4. Effect of channel concatenation number N_z on the performance of RCDNet.

N_z	N_z=0	N_z=2	N_z=8	N_z=14	N_z=20	N_z=26	N_z=32	N_z=38
PSNR	38.99	39.13	39.36	39.69	39.72	39.82	40.00	39.98
SSIM	0.9830	0.9836	0.9843	0.9851	0.9853	0.9856	0.9860	0.9859

Versions 1 and 2 (Versions 3 and 4). Furthermore, taking Version 1 as a benchmark, from Versions 2, 3, and 4, we observe that intra-stage supervision loss for rain layer plays more important role in deraining performance than that for background layer. This also reflects the effectiveness of M-net. Based on the analysis aforementioned, we thus present a Version 5 and take it as our final loss function in all subsequent experiments where we directly set $\lambda_S = \gamma_S = 1$ to make the outputs at the final stage play a dominant role, and other parameters as 0.1 to help find the correct parameter at each stage.

4.2. Network architecture

From Fig. 1, the key factors affecting our network architecture include: Resblocks number T involved in each ResNet, stage number S, and channel concatenation number N_z. As aforementioned, $\text{proxNet}_{\theta_m^{(s)}}(\cdot)$ and $\text{proxNet}_{\theta_b^{(s)}}(\cdot)$ have the same Resblocks number T and it keeps the same among all stages. In the following, we evaluate the effect of these factors on the RCDNet.

Resblocks number T at each stage. Setting S as 17 and N_z as 32, we separately select Resblocks number T as 1, 2, 3, 4, and 5. The quantitative comparison is presented in Table 2. It can be easily observed that more Resblocks usually bring higher average PSNR and SSIM. However, larger T would make gradient propagation more difficult, explaining the fact that the performance with $T = 5$ is a little inferior to the case of $T = 4$. Hence, we select T as 4 in the following experiments.

Stage number S. Table 3 and Fig. 3 clearly present the effect of stage number S on the rain removal performance of RCDNet. Here, $S = 0$ means that the initialization $\mathcal{B}^{(0)}$ is directly regraded as the recovery result. Taking $S = 0$ as a baseline, it is seen that only with 2 stages, our method achieves significant rain removal performance, which validates the essential role of the proposed M-net and B-net. We also observe that when $S = 20$, its deraining performance is slightly lower than that of $S = 17$ since larger S would make gradient propagation more difficult as explained before. Based on such observation, we easily set S as 17 throughout all our experiments.

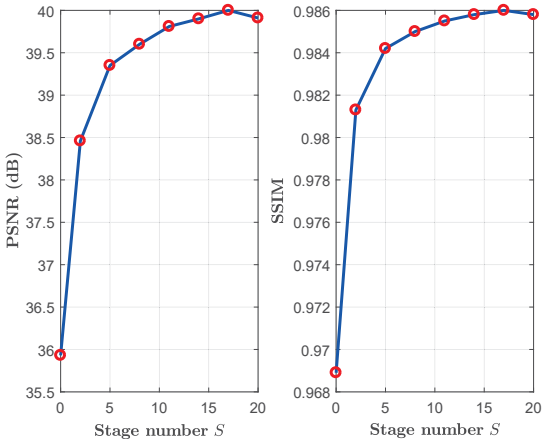

Figure 3. Average PSNR and SSIM with different stage number S.

Channel concatenation number N_z. By setting T as 4 and S as 17, Table 4 lists the PSNR and SSIM obtained by our proposed algorithm with different channel concatenation number N_z. It is easy to see that larger channel concatenation number N_z is indeed helpful for passing more effective image features and thus brings higher PSNR and SSIM. The fact confirms the rationality of the proposed concatenation operator in the network design. Besides, the case $N_z = 38$ has a litter lower performance than the case $N_z = 32$ since larger N_z with more parameters makes the network training difficult. We thus choose N_z as 32.

5. More experimental results

In this section, we demonstrate more experimental results on the datasets adopted in the main text.

Comparison methods. The comparison methods include: model-based DSC [17][5], GMM [10][6], and JCAS [5][7]; deep learning (DL)-based Clear [3][8], DDN [4][9], RESCAN [9][10], PReNet [12][11], SPANet [13][12], JORDER_E [16][13], and SIRR [15][14].

[5] https://sites.google.com/view/taixiangjiang/%E9%A6%96%E9%A1%B5/state-of-the-art-methods
[6] http://yu-li.github.io/
[7] https://sites.google.com/site/shuhanggu/home
[8] https://xueyangfu.github.io/projects/tip2017.html
[9] https://xueyangfu.github.io/projects/cvpr2017.html
[10] https://github.com/XiaLiPKU/RESCAN
[11] https://github.com/csdwren/PReNet
[12] https://stevewongv.github.io/derain-project.html
[13] https://github.com/flyywh/JORDER-E-Deep-Image-Deraining-TPAMI-2019-Journal
[14] https://github.com/wwzjer/Semi-supervised-IRR

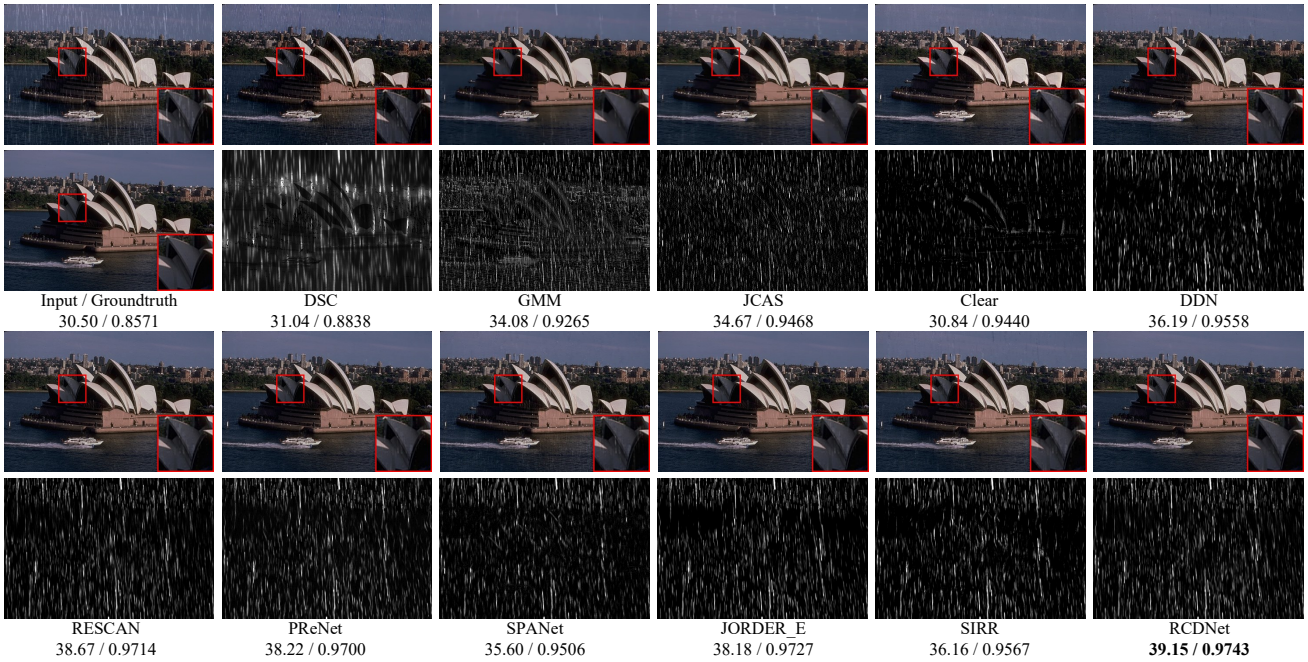

Figure 4. 1st column: input rainy image from Rain12 (upper) and groundtruth (lower). 2nd-12th column: derained results (upper) and extracted rain layers (lower) by 11 competing methods. PSNR/SSIM are listed below the corresponding images for easy reference. The best performed results are highlighted in bold. The images are better observed by zooming in on screen.

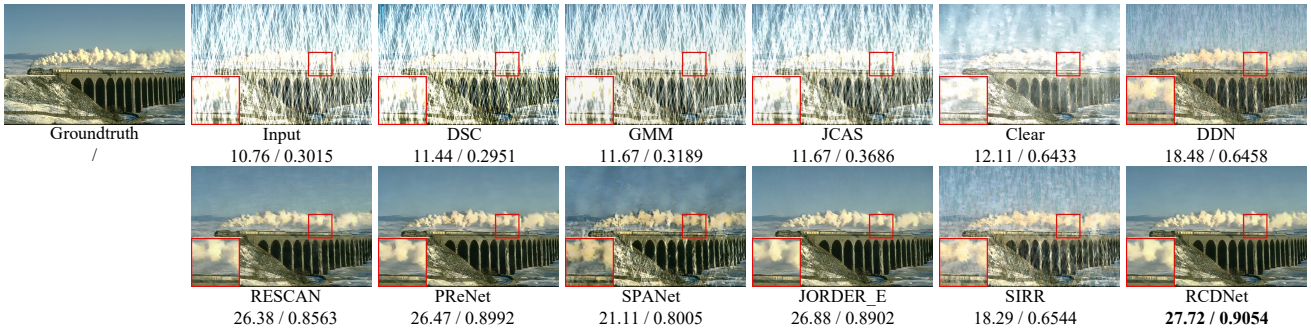

Figure 5. Rain removal performance comparisons on a rainy image from Rain100H. The images are better observed by zooming in on screen.

5.1. More results on synthetic data

Synthetic datasets. Besides Rain100L, other three frequently-used benchmark datasets are also used, including Rain100H [16], Rain1400 [4], and Rain12 [10]. In specific, Rain100H covers five types of rain streak directions and contains 1800 training image pairs and 100 testing ones. Rain1400 includes 14 kinds of different rain streak orientations and magnitudes, and consists of 12600 rainy/clean image pairs for training and 1400 ones for testing. Rain12 has only 12 pairs. Like [12], we directly apply the trained model of Rain100L on Rain12 for evaluation.

Fig. 4 illustrates the deraining performance of all competing methods on a rainy image from Rain12. As displayed, the proposed RCDNet has an evident advantage over other competing methods in rain removal and background recovery. Besides, the rain layer extracted by RCDNet contains fewer unexpected background details than that by other competing methods.

Fig. 5 and Fig. 6 depict the rain removal comparison results on the other two rainy images, selected from Rain100H and Rain1400, respectively. From the two figures, we can easily conclude that as compared with other competing methods, our proposed RCDNet not only removes more rain streaks but also preserves background details better. Even the two input rainy images have very different rain patterns, our method still obtains the best PSNR and SSIM for both of them.

Table 5 reports the quantitative results of all competing methods. It is seen that our RCDNet attains best deraining performance among all competing methods on each dataset. Moreover, even we only adopt the single loss as the ob-

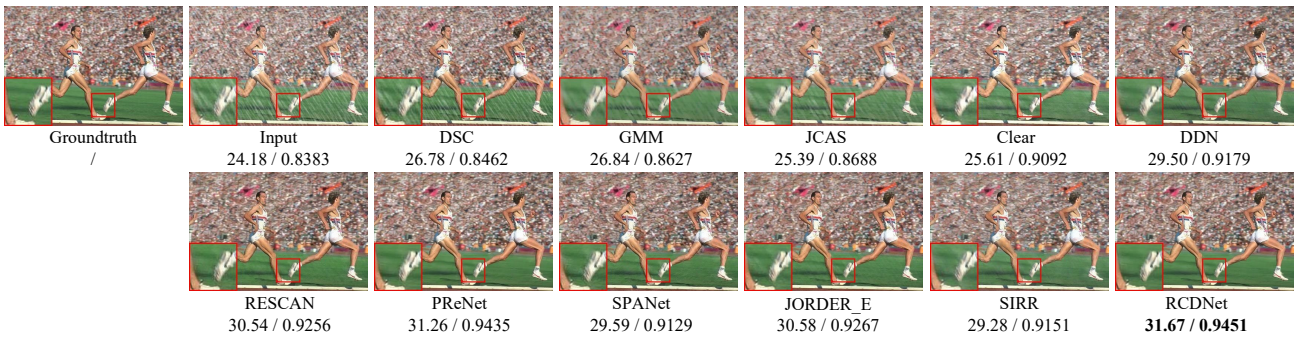

Figure 6. Rain removal performance comparisons on a rainy image from Rain1400. The best performed results are highlighted in bold. The images are better observed by zooming in on screen.

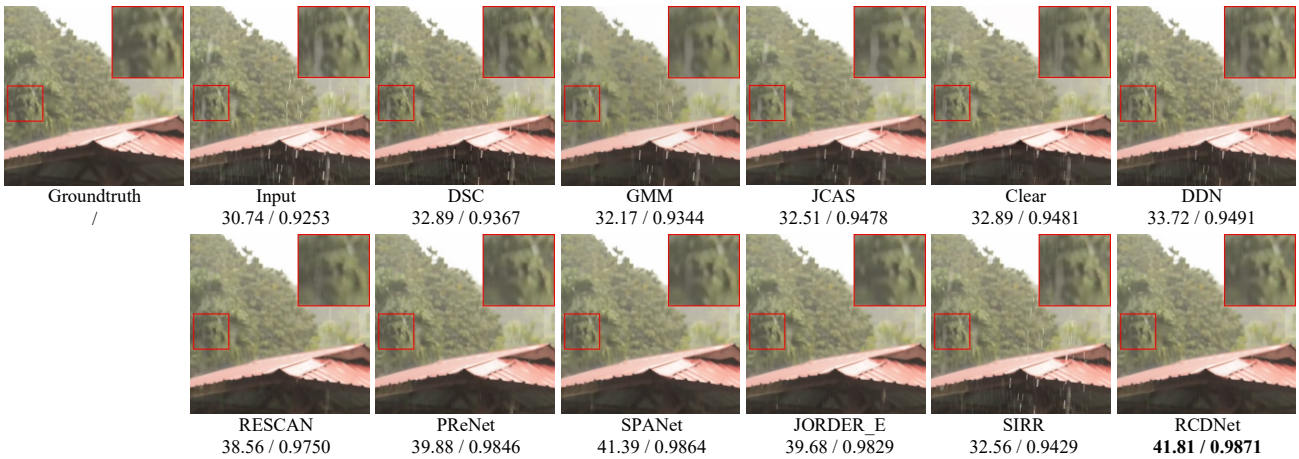

Figure 7. Rain removal performance comparisons on a real rainy image with various (long/thin/heavy) rain streaks from SPA-Data. The images are better observed by zooming in on screen.

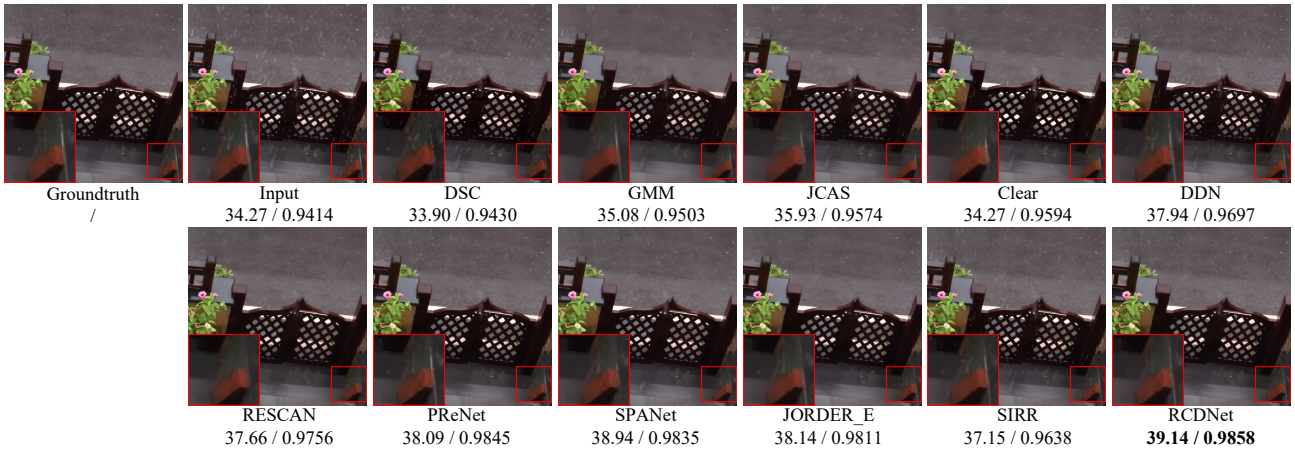

Figure 8. Rain removal performance comparisons on a real rainy image with light rain streaks from SPA-Data. The images are better observed by zooming in on screen.

jective function, that is, Version 1 in Table 1, our network RCDNet$_1$ still has a dominant deraining performance.

5.2. More results on real data

Real datasets. We then analyze the performance of all competing methods on two real datasets: the first one (called SPA-Data) from [13] contains 638492 rainy/clean image pairs for training and 1000 testing ones, and the second one (called Internet-Data) from [13, 15] includes 147 rainy images without groundtruth. Specifically, the SPA-Data is semi-automatically generated and the Internet-Data is collected from Internet and includes many hard samples with complicated rain streaks.

Fig. 7 and Fig. 8 show two real rainy images with very

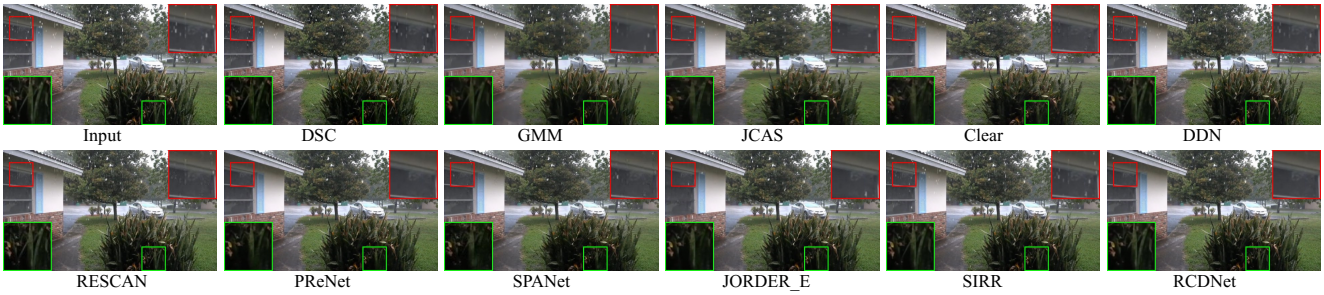

Figure 9. Derained results for one sample with complicated rain type from Internet-Data. The images are better observed by zooming in on screen.

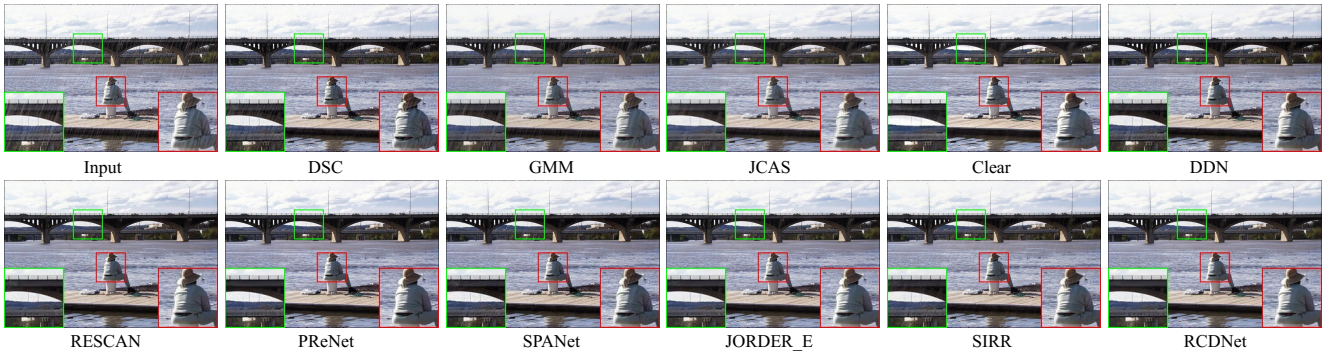

Figure 10. Derained results for one sample with dense long rain streaks from Internet-Data. The images are better observed by zooming in on screen.

Table 5. PSNR and SSIM comparisons on four benchmark datasets. The best and second best results on each dataset are indicated by bold and bold italic, respectively.

Datasets	Rain100L		Rain100H		Rain1400		Rain12	
Metrics	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Input	26.90	0.8384	13.56	0.3709	25.24	0.8097	30.14	0.8555
DSC[17]	27.34	0.8494	13.77	0.3199	27.88	0.8394	30.07	0.8664
GMM[10]	29.05	0.8717	15.23	0.4498	27.78	0.8585	32.14	0.9145
JCAS[5]	28.54	0.8524	14.62	0.4510	26.20	0.8471	33.10	0.9305
Clear[3]	30.24	0.9344	15.33	0.7421	26.21	0.8951	31.24	0.9353
DDN[4]	32.38	0.9258	22.85	0.7250	28.45	0.8888	34.04	0.9330
RESCAN[9]	38.52	0.9812	29.62	0.8720	32.03	0.9314	36.43	0.9519
PReNet[12]	37.45	0.9790	30.11	***0.9053***	32.55	***0.9459***	36.66	0.9610
SPANet[13]	35.33	0.9694	25.11	0.8332	29.85	0.9148	35.85	0.9572
JORDER_E[16]	38.59	0.9834	30.50	0.8967	32.00	0.9347	36.69	0.9621
SIRR[15]	32.37	0.9258	22.47	0.7164	28.44	0.8893	34.02	0.9347
$RCDNet_1$	***39.90***	***0.9855***	***30.91***	0.9037	***32.78***	0.9446	***37.63***	***0.9636***
RCDNet	**40.00**	**0.9860**	**31.28**	**0.9093**	**33.04**	**0.9472**	**37.71**	**0.9649**

different rain patterns from SPA-Data to comprehensively evaluate the rain removal performance of all competing methods, both visually and quantitatively. As observed, traditional model-based DSC, GMM, and JCAS leave obvious rain streaks in the derained results. Among DL-based methods, some cannot perfectly remove all rain streaks such as DDN, PReNet, JORDER_E, and SIRR, while some blur image details, including Clear and SPANet. For RESCAN, it leaves some rain streaks as shown in Fig. 7 while adversely loses background textures as presented in Fig. 8. Our RCDNet, however, can always achieve an evident superior performance than other methods under different rain types.

Table 6. PSNR and SSIM comparisons on SPA-Data [13].

Methods	Input	DSC	GMM	JCAS	Clear	DDN	RESCAN
PSNR	34.15	34.95	34.30	34.95	34.39	36.16	38.11
SSIM	0.9269	0.9416	0.9428	0.9453	0.9509	0.9463	0.9707
Methods	PReNet	SPANet	JORDER_E	SIRR	$RCDNet_1$	RCDNet	/
PSNR	40.16	40.24	40.78	35.31	***40.99***	**41.47**	/
SSIM	***0.9816***	0.9811	0.9811	0.9411	***0.9816***	**0.9834**	/

Table 6 compares the derained results on SPA-Data of all competing methods quantitatively. It is easy to see that even for such complex and diverse rain patterns, the proposed $RCDNet_1$ with the simplest loss function still significantly outperforms other comparison methods.

Further, we select two hard samples with various rain densities from Internet-Data (different from the rain types in the main text) to evaluate the generalization ability of all comparison methods. From Fig. 9 and Fig. 10, we can observe that traditional model-based methods always leave evident rain streaks. Although DL-based ones remove obvious rain streaks, they still leave distinct rain marks or blur some background details. Comparatively, the proposed RCDNet has a better generalization ability as it can better preserve image textures as well as more sufficiently remove rain streaks for these unseen complex rain types.

6. Model verification on more datasets

In this section, we provide more deraining results of RCDNet based on more samples with diverse rain patterns involved in the datasets above (besides Rain100L in the main

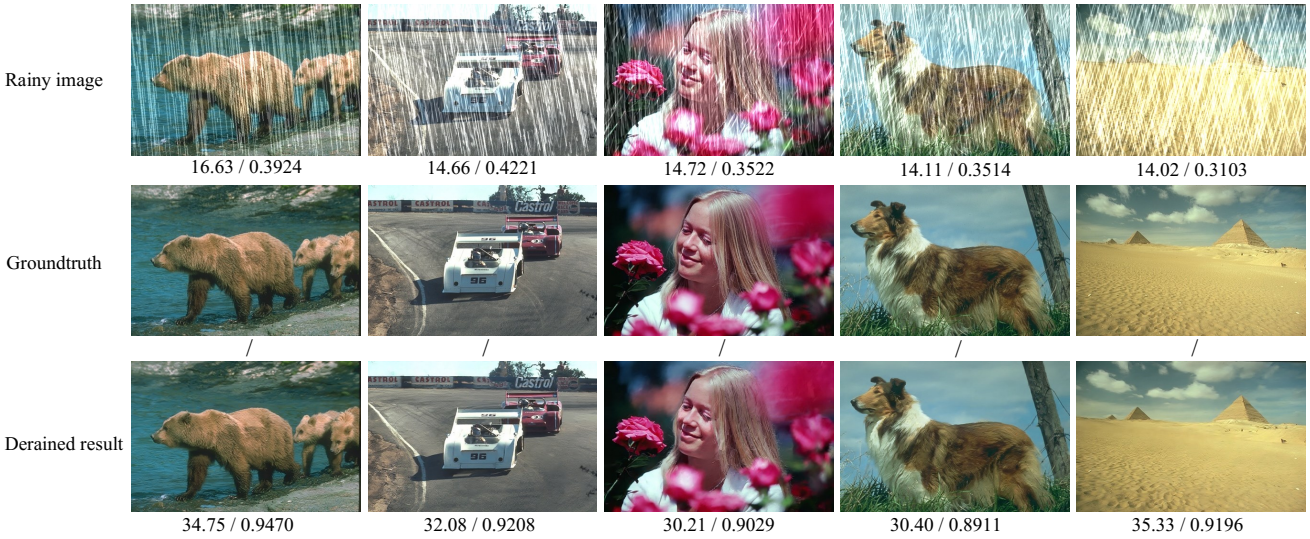

Figure 11. (a) The first row: 5 representative rainy images with different rain densities and directions involved in Rain100H. (b) The second row: the corresponding groundtruth. (c) The third row: The derained results of our RCDNet. PSNR/SSIM are listed below the corresponding images for easy reference.

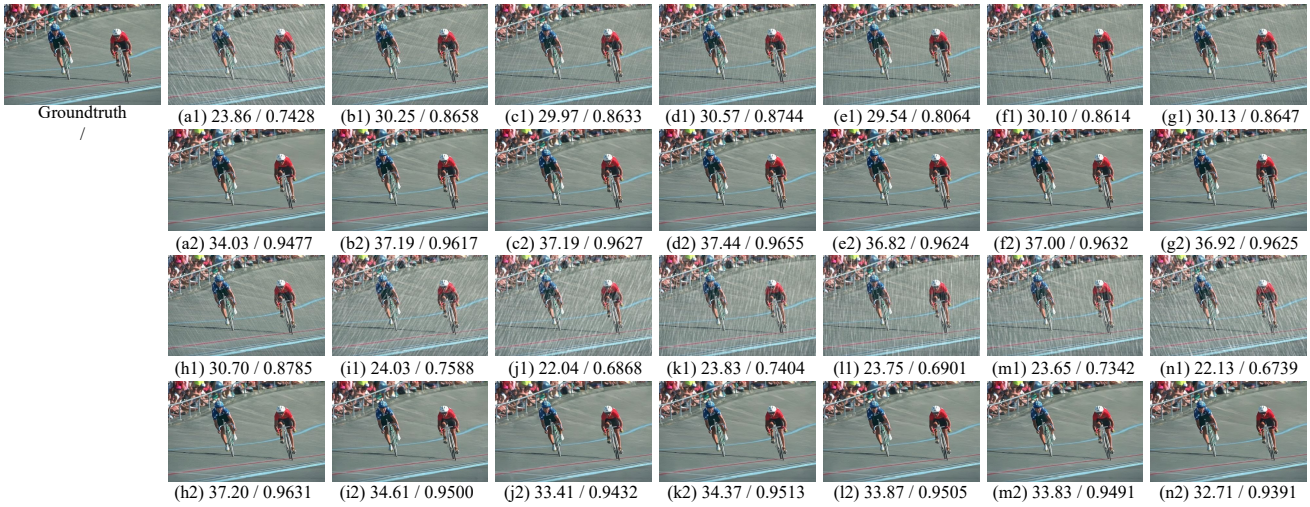

Figure 12. (a1)-(n1) The input rainy images with 14 kinds of different rain streak orientations and magnitudes from Rain1400. (a2)-(n2) The corresponding derained results of the RCDNet. The 14 inputs share the same groundtruth as displayed in the first column. PSNR/SSIM are listed below the corresponding images for easy reference. The images are better observed by zooming in on screen.

text), and visualize the learned rain kernels, so as to fully verify the mechanism of the proposed RCDNet.

For the three datasets, including Rain100H with 5 types of rain streaks, Rain1400 with 14 kinds of ones, and SPA-Data, we correspondingly select several rainy images with representative rain patterns as shown in Fig. 11, Fig. 12, and Fig. 13, respectively. From the visual and quantitative demonstration, we can easily observe that even for such complex rain patterns, RCDNet always attains excellent deraining performance. This substantiates the good flexibility and generality of our method. Besides, the corresponding rain kernels for these datasets extracted by our methods are shown in Fig. 14. Clearly, it is observed that the shapes of rain kernels are different among the three datasets, and the rain kernels learned from Rain1400 and SPA-Data are more diverse than that learned from Rain100H. This fits perfectly with the diversity of these datasets. It confirms that our method is indeed capable of automatically learning diverse rain kernels that are potentially useful for the related tasks on rainy images.

References

[1] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009. 1

[2] David L Donoho. De-noising by soft-thresholding. *IEEE*

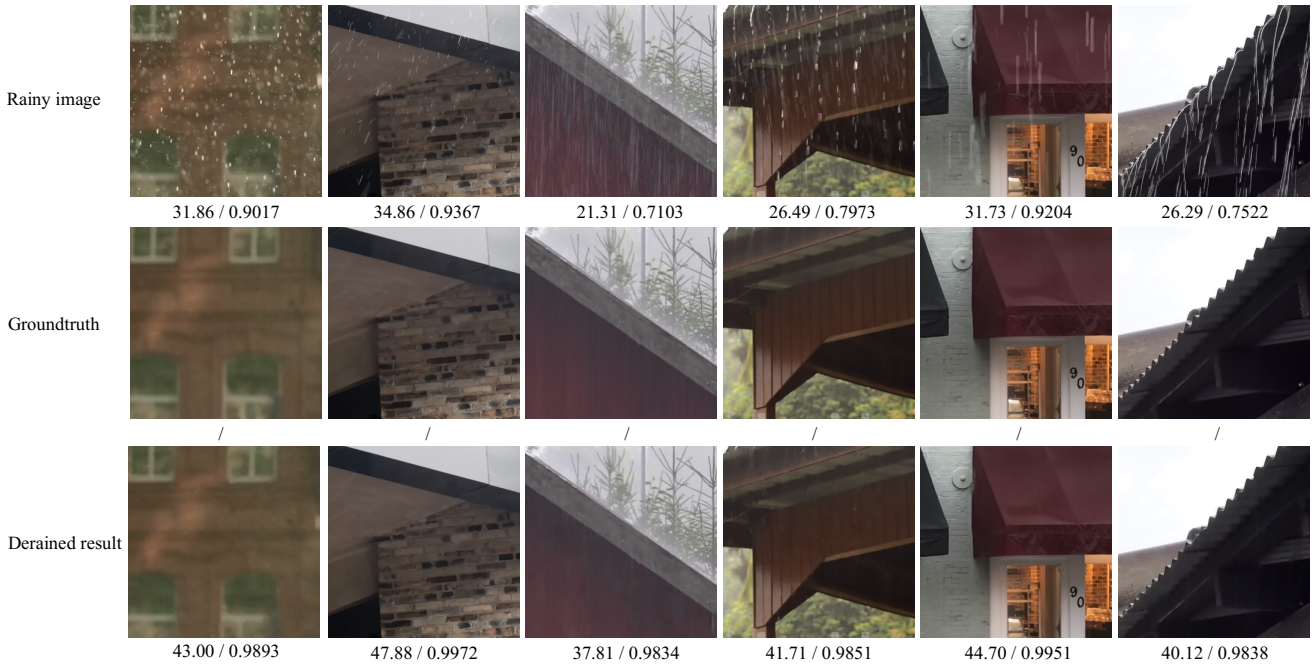

Figure 13. (a) The first row: 6 rainy images representing different rain patterns from small raindrops to long/wide/thick strips involved in SPA-Data. (b) The second row: the corresponding groundtruth. (c) The third row: The derained results of our RCDNet. PSNR/SSIM are listed below the corresponding images for easy reference. The images are better observed by zooming in on screen.

(a) Rain100H (b) Rain1400 (c) SPA-Data

Figure 14. Rain kernels $\mathcal{C} = \{\mathcal{C}_n\}_{n=1}^{N=32}$ learned from Rain100H, Rain1400, and SPA-Data, respectively, by the proposed method.

transactions on information theory, 41(3):613–627, 1995. 1

[3] Xueyang Fu, Jiabin Huang, Xinghao Ding, Yinghao Liao, and John Paisley. Clearing the skies: A deep network architecture for single-image rain removal. *IEEE Transactions on Image Processing*, 26(6):2944–2956, 2017. 4, 7

[4] Xueyang Fu, Jiabin Huang, Delu Zeng, Huang Yue, Xinghao Ding, and John Paisley. Removing rain from single images via a deep detail network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3855–3863, 2017. 4, 5, 7

[5] Shuhang Gu, Deyu Meng, Wangmeng Zuo, and Zhang Lei. Joint convolutional analysis and synthesis sparse representation for single image layer separation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1708–1716, 2017. 4, 7

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[7] Q. Huynh-Thu and M. Ghanbari. Scope of validity of p-snr in image/video quality assessment. *Electronics Letters*, 44(13):800–801, 2008. 3

[8] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Computer Science*, 2014. 3

[9] Xia Li, Jianlong Wu, Zhouchen Lin, Hong Liu, and Hongbin Zha. Recurrent squeeze-and-excitation context aggregation net for single image deraining. In *Proceedings of the European Conference on Computer Vision*, pages 254–269, 2018. 4, 7

[10] Yu Li. Rain streak removal using layer priors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2736–2744, 2016. 4, 5, 7

[11] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 3

[12] Dongwei Ren, Wangmeng Zuo, Qinghua Hu, Pengfei Zhu, and Deyu Meng. Progressive image deraining networks: a better and simpler baseline. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3937–3946, 2019. 4, 5, 7

[13] Tianyu Wang, Xin Yang, Ke Xu, Shaozhe Chen, Qiang Zhang, and Rynson WH Lau. Spatial attentive single-image deraining with a high quality real rain dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12270–12279, 2019. 4, 6, 7

[14] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Processing*, 13(4):600–612, 2004. 3

[15] Wei Wei, Deyu Meng, Qian Zhao, Zongben Xu, and Ying Wu. Semi-supervised transfer learning for image rain removal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3877–3886, 2019. 4, 6, 7

[16] Wenhan Yang, Robby T. Tan, Jiashi Feng, Jiaying Liu, Shuicheng Yan, and Zongming Guo. Joint rain detection and removal from a single image with contextualized deep networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2019. 3, 4, 5, 7

[17] Luo Yu, Xu Yong, and Ji Hui. Removing rain from a single image via discriminative sparse coding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3397–3405, 2015. 4, 7