A Model-driven Deep Neural Network for Single Image Rain Removal

Hong Wang¹,*; Qi Xie¹,*; Qian Zhao¹; Deyu Meng²,†
¹Xi’an Jiaotong University; ²Macau University of Science and Technology
{hongwang01,xq.liwu}@stu.xjtu.edu.cn timmy.zhaqian@gmail.com dymeng@mail.xjtu.edu.cn

Abstract

Deep learning (DL) methods have achieved state-of-the-art performance in the task of single image rain removal. Most of current DL architectures, however, are still lack of sufficient interpretability and not fully integrated with physical structures inside general rain streaks. To this issue, in this paper, we propose a model-driven deep neural network for the task, with fully interpretable network structures. Specifically, based on the convolutional dictionary learning mechanism for representing rain, we propose a novel single image deraining model and utilize the proximal gradient descent technique to design an iterative algorithm only containing simple operators for solving the model. Such a simple implementation scheme facilitates us to unfold it into a new deep network architecture, called rain convolutional dictionary network (RCDNet), with almost every network module one-to-one corresponding to each operation involved in the algorithm. By end-to-end training the proposed RCDNet, all the rain kernels and proximal operators can be automatically extracted, faithfully characterizing the features of both rain and clean background layers, and thus naturally lead to its better deraining performance, especially in real scenarios. Comprehensive experiments substantiate the superiority of the proposed network, especially its well generality to diverse testing scenarios and good interpretability for all its modules, as compared with state-of-the-arts both visually and quantitatively.

1. Introduction

Images taken under various rain conditions often suffer from unfavorable visibility, and always severely affect the performance of outdoor computer vision tasks, such as object tracking [5], video surveillance [37], and pedestrian detection [31]. Hence, removing rain streaks from rainy images is an important pre-processing task and has drawn much research attention in the recent years [39, 26].

In the past years, various methods have been proposed for single image rain removal task. Many researchers made

¹Corresponding author
*Equal contribution
background layer, and introduced various prior structures to regularize and separate them. Along this research line, the representative methods include layer priors with Gaussian mixture model (GMM) [28], discriminative sparse coding (DSC) [51], and joint convolutional analysis and synthesis sparse representation (JCAS) [13]. Especially, inspired by the fact that rain streaks repeatedly appear at different locations over a rainy image with similar local patterns like shape, thickness, and direction, very recently researchers represented this configuration of rain layer by the convolutional dictionary learning model [15, 16]. Such a representation finely delivers this prior knowledge by imposing rain kernels (conveying repetitive local patterns) on sparse rain maps, as intuitively depicted in Fig. 1 (a). These methods thus achieved state-of-the-art (SOTA) performance when the background can also be well represented, e.g., by low-rank prior in surveillance video sequences [25].

Albeit effective in certain applications, the rationality of these techniques depends on the subjective prior assumptions imposed on the unknown background and rain layers to be recovered. In real scenarios, however, such learning regimes could not always adapt to different rainy images with complex, diverse, and variant structures collected from different resources. Besides, these methods generally need time-consuming iterative computations, often with efficiency issue in real applications.

Driven by the significant success of deep learning (DL) in low level vision, recent years have also witnessed the rapid progress of deep convolutional neural networks (CNN) for single image rain removal [8, 52, 53, 40]. The current DL-based derainers mainly focus on designing network modules, and then train network parameters based on abundant rainy/clean image pairs to extract the background layer. Typical deraining network structures include deep detail network (DDN) [9], recurrent squeeze-and-excitation context aggregation module (RESCAN) [27], progressive image deraining network (PRNet) [35], spatial attentive unit (SPANet) [41], and many others.

These DL strategies, however, also possess evident deficiencies. The most significant one is their weak interpretability. Network structures are always complicated and diverse, making it difficult to analyze the role of different modules and understand the underlying insights of their mechanism. Besides, most of them treat CNN as an encapsulated end-to-end mapping module without deepening into the rationality, and neglect the intrinsic prior knowledge of rain streaks such as sparsity and nonlocal similarity. This makes this methodology easily trapped into the overfitting-to-training-sample issue.

To alleviate the aforementioned issues, this paper designs an interpretable deep network, which sufficiently considers the characteristics of rain streaks and attempts to combine the advantages of the conventional model-driven prior-based and current data-driven DL-based methodologies. Specifically, our contributions are mainly three-fold:

Firstly, we propose a concise rain convolutional dictionary (RCD) model for single image by exploiting the intrinsic convolutional dictionary learning mechanism to encode rain shapes, and specifically adopt the proximal gradient technique [2] to design an optimization algorithm for solving it. Different from traditional solvers for the RCD model containing complex operations (e.g., Fourier transformation), the algorithm only contains simple computations (see Fig. 1 (b)) easy to be implemented by general network modules. This facilitates our algorithm capable of being easily unfolded into a deep network architecture.

Secondly, by unfolding the algorithm, we design a new deep network architecture for image deraining, called RCD-Net. The specificity of this network lies on its exact step-by-step corresponding relationship between its modules and the algorithm operators, and thus successively possesses the interpretability of all its modules as that of all steps in the algorithm. Specifically, as shown in Fig. 1 (b) and (c), each iteration of the algorithm contains two sub-steps, respectively updating the rain map (convoluted by the learned rain kernels) and background layer, and each stage of the RCDNet also contains two sub-networks (M-net and B-net). Each output of the intermediate layer in the network is thus with clear interpretation, which greatly facilitates a deeper analysis on what happens inside the network during training, and a comprehensive understanding why the network works or not (as the analysis presented in Sec. 5.2).

Thirdly, comprehensive experimental results substantiate the superiority of the RCDNet beyond SOTA conventional prior-based and current DL-based methods both quantitatively and visually. Especially, attributed to its well interpretability, not only the underlying rationality and insights of the network can be intuitively understood through visualizing the amelioration process (like the gradually rectified background and rain maps) over all network layers by general users, but also the network can yield generally useful rain kernels for expressing rain shapes and proximal operators for delivering the prior knowledge of background and rain maps for a rainy image, facilitating their general availability to more real-world rainy images.

The paper is organized as follows. Sec. 2 reviews the related rain removal work. Sec. 3 presents the RCD model for rain removal as well as the algorithm designed for solving it. Then Sec. 4 introduces the unfolding deep network for the algorithm. The experimental results are demonstrated in Section 5 and the paper is finally concluded.

2. Related work

In this section, we give a brief review on the most related work on rain removal for images. Depending on the input data, the existing algorithms can be categorized into two
groups: video based and single image based ones.

2.1. Video deraining methods

Garg and Nayar [10] first tried to analyze the visual effects of raindrops on imaging systems, and utilized a space-time correlation model to capture the dynamics of raindrops and a physics-based motion blur model to illustrate the photometry of rain. For better visual quality, they further proposed to increase the exposure time or reduce the depth of field of a camera [12, 11]. Later, both temporal and chromatic properties of rain were considered and then background layer was extracted from rainy video by utilizing different strategies such as K-means clustering [55], Kalman filter [33], and GMM [3]. Besides, a spatio-temporal frequency based raindrop detection method was provided in [1].

In recent years, researchers introduced more intrinsic characteristics of rainy video to the task, e.g., similarity and repeatability of rain streaks [4], low-rankness among multi-frames [20], and sparsity and smoothness of rain streaks [18]. To handle heavy rain and dynamic scenes, a matrix decomposition based video deraining algorithm was presented in [36]. Afterwards, rain streaks were encoded as a patch based GMM to adapt a wider range of rain variations [45]. More characteristics of rain streaks in a rainy video were explored including repetitive local patterns and multi-scale configurations and they were described as multi-scale convolutional sparse coding model [25]. More recently, there are some DL-based methods proposed for this task. Chen et al. [19] presented a CNN architecture and utilized superpixel to handle torrential rain fall with opaque streak occlusions. To further improve visual quality, Liu et al. [30] designed a joint recurrent rain removal and reconstruction network that integrated rain degradation classification, rain removal, and background details reconstruction. To handle dynamic video contexts, they further developed a dynamic routing residue recurrent network [29]. Though these methods work well for videos, they cannot directly perform in single image cases due to the lack of temporal knowledge.

2.2. Single image deraining methods

Compared with video deraining task under a sequence of images, rain removal from a single image is much more challenging. The early attempts utilized the model-driven strategies by decomposing a single rainy image into low frequency part (LFP) and high frequency part (HFP) and then specifically extracted rain layer from the HFP based on various processing such as guided filter [6, 21] and nonlocal means filtering [23]. Later, researchers made more focus on exploring the prior knowledge of rain and rain-free layers of a rainy image, and designing proper regularizer to extract and separate them [22, 38, 51, 28, 42, 56]. E.g., [13] considered the specific sparsity characteristics of rain-free and rain parts and expressed them as the joint analysis and synthesis sparse representation models, respectively. [15] used a similar manner to deliver local repetitive patterns of rain streaks across the image as the RCD model. Albeit achieving good performance on certain scenarios, these prior-based methods rely on the subjective prior assumptions, while could not always generally work well for practical complicated and highly diverse rain shapes in real rainy images collected from different resources.

Recently, a number of DL-based single image rain streak removal methods were proposed through constructing diverse network modules [8, 9, 27, 52, 53]. To handle heavy rain, Yang et al. [49] developed a multi-stage joint rain detection and estimation network for single image (JORDER_E). Very recently, Ren et al. [35] designed a PReNet that repeatedly unfolded several Resblocks and an LSTM layer. Wang et al. [41] presented an attention unit based SPANet for removing rain in a local-to-global manner. Through using abundant rainy/clean image pairs to train the deep model, these methods achieve favorable visual quality and SOTA quantitative measures of derained results. Most of these methods, however, just utilize network modules assembled with some off-the-shelf components in current DL toolkits to directly learn background layer in an end-to-end way, and largely ignore the intrinsic prior structures inside the rain streaks. This makes them lack of evident interpretability in their network architectures and still have room for further performance enhancement.

At present, there is a new type of single image derainers that try to combine prior and DL methodologies. For example, Mu et al. [32] utilized CNN to implicitly learn prior knowledge for background and rain streaks, and formulated them into traditional bi-layer optimization iterations. Wei et al. [44] provided a semi-supervised rain removal method (SIRR) that described rain layer prior as a general GM-M and jointly trained the backbone–DDN. Albeit obtaining initial success, they still use CNN architectures as their main modules to construct the network, which is thus still lack of sufficient interpretability.

3. RCD model for single image deraining

3.1. Model formulation

For a observed color rainy image denoted as \( O \in \mathbb{R}^{H \times W \times 3} \), where \( H \) and \( W \) are the height and width of the image, respectively, it can be rationally separated as:

\[
O = B + R,
\]

where \( B \) and \( R \) represent the background and rain layers of the image, respectively. Then, the aim of most of DL-based deraining methods is to estimate the mapping function (expressed by a deep network) from \( O \) to \( B \) (or \( R \)).
Instead of heuristically constructing a complex deep network architecture, we first consider the problem under the conventional prior-based methodology through exploiting the prior knowledge for representing rain streaks [13, 15, 25]. Specifically, as shown in Fig. 1 (a), by adopting the RCD mechanism, the rain layer can be modeled as:

\[
R^c = \sum_{n=1}^{N} C^c_n \otimes M_n, c = 1, 2, 3,
\]

where \(R^c\) denotes the \(c^{th}\) color channel of \(R\), and \(\{C^c_n\}_{n,c} \subset \mathbb{R}^{k \times k}\) is a set of rain kernels which describes the repetitive local patterns of rain streaks, and \(\{M_n\}_{n} \subset \mathbb{R}^{H \times W}\) represents the corresponding rain maps representing the locations where local patterns repeatedly appear. \(N\) is the number of kernels and \(\otimes\) is the 2-dimensional (2D) convolutional operation. For conciseness, we rewrite (2) as \(R = \sum_{n=1}^{N} C_n \otimes M_n\) throughout the paper, where \(C_n \in \mathbb{R}^{k \times k \times 3}\) is the tensor form of \(C^c_n\) and the convolution is performed between \(C_n\) and the matrix \(M_n\) one channel by one channel. Then, we can rewrite the model (1) as:

\[
O = B + \sum_{n=1}^{N} C_n \otimes M_n.
\]

It should be noted that the rain kernels actually can be viewed as a set of convolutional dictionary [16] for representing repetitive and similar local patterns underlying rain streaks, and a small number of rain kernels can reasonably represent wide range of rain shapes\(^1\). They are common knowledge for representing different rain types across all rainy images, and thus could be learned from abundant training data by virtue of the strong learning capability of end-to-end training manner of deep learning (see more details in Sec. 4). Unlike rain kernels, the rain maps must vary with the input rainy image as the locations of rain streaks are totally random. Therefore, for predicting the clean image from a testing input rainy one, the key issue is to output \(M_n\)s and \(B\) from \(O\) with the rain kernels \(C_n\)s fixed, and the corresponding optimization problem is:

\[
\min_{\mathcal{M}, \mathcal{B}} \left\| O - B - \sum_{n=1}^{N} C_n \otimes M_n \right\|^2_F + \alpha g_1(\mathcal{M}) + \beta g_2(\mathcal{B}),
\]

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

### 3.2. Optimization algorithm

Since we want to build a possibly perfect step-by-step corresponding deep unfolding network architecture for solving the problem (4), it is critical to build an algorithm which contains only simple computations easy to be transformed to network modules. The traditional solvers for RCD-based model usually contain certain complicated operations, e.g., the Fourier transform and inverse Fourier transform [16, 46, 25], which are hard to accomplish such exact transformation from algorithm to network structure. We thus prefer to build a new algorithm for solving the problem through alternately updating \(\mathcal{M}\) and \(\mathcal{B}\) by proximal gradient method [2]. In this manner, only simple computations can be involved. The details are as follows:

**Updating \(\mathcal{M}\):** The rain maps \(\mathcal{M}\) can be updated by solving the quadratic approximation (2) of the problem (4) as:

\[
\min_{\mathcal{M}} \frac{1}{2} \left\| \mathcal{M} - (\mathcal{M}^{(s-1)} - \eta_1 \nabla f (\mathcal{M}^{(s-1)}) \right\|^2_F + \alpha \eta_1 g_1 (\mathcal{M}),
\]

where \(\mathcal{M}^{(s-1)}\) is the updating result of the last iteration, \(\eta_1\) is the stepsize parameter, and \(f (\mathcal{M}^{(s-1)}) = \left\| \mathcal{O} - B^{(s-1)} - \sum_{n=1}^{N} C_n \otimes M_n^{(s-1)} \right\|^2_F\). Corresponding to general regularization terms [7], the solution of Eq. (5) is:

\[
\mathcal{M}^{(s)} = \text{prox}_{\alpha \eta_1} (\mathcal{M}^{(s-1)} - \eta_1 \nabla f (\mathcal{M}^{(s-1)})). \tag{6}
\]

Moreover, by substituting

\[
\nabla f (\mathcal{M}^{(s-1)}) = C \otimes T \left( \sum_{n=1}^{N} C_n \otimes M_n^{(s-1)} + B^{(s-1)} - \mathcal{O} \right), \tag{7}
\]

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

\[
\mathcal{M}^{(s)} = \text{prox}_{\alpha \eta_1} \left( \mathcal{M}^{(s-1)} - \eta_1 C \otimes T ( \sum_{n=1}^{N} C_n \otimes M_n^{(s-1)} + B^{(s-1)} - \mathcal{O} ) \right). \tag{8}
\]

where \(\text{prox}_{\alpha \eta_1} (\cdot)\) is the proximal operator dependent on the regularization term \(g_1(\cdot)\) with respect to \(\mathcal{M}\). Instead of choosing a fixed regularizer in the model, the form of the proximal operator can be automatically learned from training data. More details will be presented in the next section.

**Updating \(\mathcal{B}\):** Similarly, the quadratic approximation of the problem (4) with respect to \(\mathcal{B}\) is:

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

where \(\nabla h (\mathcal{B}^{(s-1)}) = \sum_{n=1}^{N} C_n \otimes M_n^{(s-1)} + B^{(s-1)} - \mathcal{O}\), and it is easy to deduce that the final updating rule for \(\mathcal{B}\) is\(^3\):

\[
\mathcal{B}^{(s)} = \text{prox}_{\beta \eta_2} \left( 1 - \eta_2 B^{(s-1)} + \eta_2 \mathcal{O} - \sum_{n=1}^{N} C_n \otimes M_n^{(s)} \right). \tag{10}
\]

\(^2\)For any tensor \(A \in \mathbb{R}^{H \times W \times N \times 3}\), we can calculate the \(n^{th}\) channel of \(C \otimes T A\) by \(\sum_{c=1}^{3} C_{(i,i,n,c)} \otimes T A_{(i,i,c)}\).

\(^3\)It can be proved that, with small enough \(\eta_1\) and \(\eta_2\), Eq. (8) and Eq. (10) can both lead to the reduction of objective function (4) [2].
where prox_{βη_2} (·) is the proximal operator correlated to the regularization term η_2(·) with respect to B.

Based on this iterative algorithm, we can then construct our deep unfolding network as follows.

4. The rain convolutional dictionary network

Inspired by the recently raised deep unfolding techniques in various tasks such as deconvolution [54], compressed sensing [50], and dehazing [48], we build a network structure for single image rain removal task by unfolding each iterative steps of the aforementioned algorithm as the corresponding network module. We especially focus on making all network modules one-to-one corresponding to the algorithm implementation operators, for better interpretability.

As shown in Fig. 2 (a), the proposed network consists of S stages, corresponding to S iterations of the algorithm for solving (4). Each stage achieves the sequential updates of M and B by M-net and B-net. As displayed in Fig. 2 (b), exactly corresponding to each iteration of the algorithm, in each stage of the network, M-net takes the observed rainy image O and the previous outputs B^{(s-1)} and M^{(s-1)} as inputs, and outputs an updated M^{(s)}; and then B-net takes O and M^{(s)} as inputs, and outputs an updated B^{(s)}.

4.1. Network design

The key issue of unrolling the algorithm here is how to represent the two proximal operators involved in (8) and (10) while other operations can be naturally performed with generally used operators in normal networks [34]. In this work, we simply choose a ResNet [14] to construct the two proximal operators as many other works did [47, 48]. Then, we can separately decompose the updating rules for M as (8) and B as (10) into sub-steps and achieve the following procedures for the s-th stage of the RCDNet:

\[
\begin{align*}
\hat{R}^{(s)} &= O - B^{(s-1)}, \\
\tilde{R}^{(s)} &= \sum_{n=1}^{N} C_n \otimes M_n^{(s-1)}, \\
E^{(s)} &= \eta_1 C \odot T \left( \tilde{R}^{(s)} - \hat{R}^{(s)} \right), \\
M^{(s)} &= \text{proxNet}_{\theta_m} \left( M^{(s-1)} - E^{(s)} \right), \\
\hat{R}^{(s)} &= \sum_{n=1}^{N} C_n \otimes M_n^{(s)}, \\
\tilde{R}^{(s)} &= O - \hat{R}^{(s)}, \\
B^{(s)} &= \text{proxNet}_{\theta_b} \left( (1 - \eta_2) B^{(s-1)} + \eta_2 \tilde{R}^{(s)} \right),
\end{align*}
\]

where \text{proxNet}_{\theta_m} (·) and \text{proxNet}_{\theta_b} (·) are two ResNets consisting of several Resblocks with the parameters \theta_m^{(s)} and \theta_b^{(s)} at the s-th stage, respectively.

We can then design the network architecture, as shown in Fig. 2, by transforming the operators in (11) and (12) step-by-step. All the parameters involved 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 C, \eta_1, and \eta_2.

It should be indicated that both of the two sub-networks are very interpretable. As shown in Fig. 2 (b), the M-net accomplishes the extraction of residual information \mathcal{E}^{(s)} of rain maps. Specifically, \hat{R}^{(s)} is the rain layer estimated with the previous background \hat{B}^{(s-1)}, and \tilde{R}^{(s)} is the rain layer achieved by the generative model (2) with the estimated \hat{M}^{(s-1)}. Then the M-net calculates the residual information between the two rain layers obtained in this two ways, and extracts the residual information \mathcal{E}^{(s)} of rain maps with the transposed convolution of rain kernels to update the rain map. Next, the B-net recovers the background \tilde{B}^{(s)} estimated with current rain kernel and rain maps \hat{M}^{(s)}, and fuses this estimated \tilde{B}^{(s)} with the previously estimated \hat{B}^{(s-1)} by
weighted parameters $\eta_2$ and $(1-\eta_2)$ to get the updated background $B'(s)$. Here, we set $M(0)$ as 0 and initialize $B(0)$ by a convolutional operator on $O^4$.

**Remark:** From Fig. 2, the input tensor of proxNet$_{df^3}(\cdot)$ has the same size $H \times W \times 3$ as the to-be-estimated $B$. Evidently, this is not beneficial for learning $B$ since most of the previous updating information would be compressed due to few channels. To better keep and deliver image features, we simply expand the input tensor at the 3rd mode for more channels in experiments (see more in supplemental file).

4.2. Network training

**Training loss.** For simplicity, we adopt the mean square error (MSE) [21] for the learned background and rain layer at every stage as the training objective function:

$$ L = \sum_{s=0}^{S} \lambda_s \left\| B'(s) - B \right\|_F^2 + \sum_{s=1}^{S} \gamma_s \left\| O - B - R(s) \right\|_F^2, \quad (13) $$

where $B'(s)$ and $R(s)$ separately denote the derained result and extracted rain layer as expressed in (12) at the $s$th stage ($s = 0, 1, \cdots, S$), $\lambda_s$ and $\gamma_s$ are tradeoff parameters.$^5$

**Implement details.** We implement our network based on a NVIDIA GeForce GTX 1080Ti GPU. We adopt the Adam optimizer [24] with the batch size of 16 and the patch size of $64 \times 64$. The initial learning rate is $1 \times 10^{-3}$ and divided by 5 every 25 epochs. The total epoch is 100.

5. Experimental results

We first conduct ablation study and model visualization to verify the underlying mechanism of the proposed network, and then present experiments on synthesized benchmark datasets and real datasets for performance evaluation.

5.1. Ablation study

**Dataset and performance metrics.** In this section, we use Rain100L to perform all the ablation studies. The synthesized dataset consists of 200 rainy/clean image pairs for training and 100 pairs for testing [49]. Two performance metrics are employed, including peak-signal-to-noise ratio (PSNR) [17] and structure similarity (SSIM) [43]. Note that 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.

Table 1 reports the effect of stage number $S$ on deraining performance of our network. Here, $S = 0$ means that the initialization $B(0)$ is directly regraded as the recovery result.$^4$

More network design details are described in supplemental file.

5.2. Model verification

We then show how the interpretability of this RCDNet facilitates an easy analysis for the working mechanism inside the network modules.

Fig. 3 presents the extracted background layer $B'(s)$ (1st row), $\hat{B}'(s)$ (2nd row) that represents the role of M-net in helping restore clean background, and rain layer $R(s)$ (3rd row) at different stages. We can find that with the increase of $s$, $R(s)$ covers more rain streaks and fewer image details, and $\hat{B}'(s)$ and $B'(s)$ are also gradually ameliorated. These should

![Figure 4](image-url)

Figure 4. At the final stage $s = 17$, the extracted rain layer, rain kernels $C_n$, and rain maps $M_n$ for the input $O$ in Fig. 3. The lower left is the rain kernels $C$ learned from Rain100L. The images are better to be zoomed in on screen.

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. Based on such observation, we easily set $S$ as 17 throughout all our experiments. More ablation results and discussions are listed in supplementary material.

**Table 1. Effect of stage number $S$ on the performance of RCDNet.**

<table>
<thead>
<tr>
<th>Stage No.</th>
<th>$S=0$</th>
<th>$S=2$</th>
<th>$S=5$</th>
<th>$S=8$</th>
<th>$S=11$</th>
<th>$S=14$</th>
<th>$S=17$</th>
<th>$S=20$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>35.93</td>
<td>38.46</td>
<td>39.35</td>
<td>39.60</td>
<td>39.81</td>
<td>39.90</td>
<td>40.00</td>
<td>39.91</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.9869</td>
<td>0.9813</td>
<td>0.9842</td>
<td>0.9850</td>
<td>0.9855</td>
<td>0.9858</td>
<td>0.9860</td>
<td>0.9858</td>
</tr>
</tbody>
</table>

Remark: From Fig. 2, the input tensor of proxNet$_{df^3}(\cdot)$ has the same size $H \times W \times 3$ as the to-be-estimated $B$. Evidently, this is not beneficial for learning $B$ since most of the previous updating information would be compressed due to few channels. To better keep and deliver image features, we simply expand the input tensor at the 3rd mode for more channels in experiments (see more in supplemental file).
Table 2. PSNR and SSIM comparisons on four benchmark datasets. Bold and bold italic indicate top 1st and 2nd rank, respectively.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Rain100L</th>
<th>Rain100H</th>
<th>Rain1400</th>
<th>Rain12</th>
</tr>
</thead>
<tbody>
<tr>
<td>metrics</td>
<td>PSNR SSIM</td>
<td>PSNR SSIM</td>
<td>PSNR SSIM</td>
<td>PSNR SSIM</td>
</tr>
<tr>
<td>Input</td>
<td>26.90 / 0.8384</td>
<td>13.56 / 0.3709</td>
<td>25.24 / 0.8097</td>
<td>30.14 / 0.8555</td>
</tr>
<tr>
<td>DSC [51]</td>
<td>27.34 / 0.8499</td>
<td>13.77 / 0.3199</td>
<td>27.88 / 0.8394</td>
<td>30.07 / 0.8664</td>
</tr>
<tr>
<td>GMM [28]</td>
<td>29.05 / 0.8717</td>
<td>15.23 / 0.4498</td>
<td>27.78 / 0.8585</td>
<td>32.14 / 0.9145</td>
</tr>
<tr>
<td>JCAS [13]</td>
<td>28.54 / 0.8524</td>
<td>14.62 / 0.4510</td>
<td>26.20 / 0.8471</td>
<td>33.10 / 0.9305</td>
</tr>
<tr>
<td>Clear [5]</td>
<td>30.24 / 0.9344</td>
<td>15.53 / 0.7421</td>
<td>26.21 / 0.8951</td>
<td>31.24 / 0.9353</td>
</tr>
<tr>
<td>DDN [9]</td>
<td>32.38 / 0.9258</td>
<td>22.85 / 0.7250</td>
<td>28.45 / 0.8888</td>
<td>34.04 / 0.9350</td>
</tr>
<tr>
<td>RESCAN</td>
<td>38.52 / 0.9812</td>
<td>29.02 / 0.8720</td>
<td>32.03 / 0.9314</td>
<td>36.43 / 0.9519</td>
</tr>
<tr>
<td>PReNet</td>
<td>37.43 / 0.9790</td>
<td>30.11 / 0.9053</td>
<td>32.55 / 0.9495</td>
<td>36.66 / 0.9610</td>
</tr>
<tr>
<td>SPANet</td>
<td>35.33 / 0.9634</td>
<td>25.11 / 0.8332</td>
<td>29.85 / 0.9148</td>
<td>35.85 / 0.9572</td>
</tr>
<tr>
<td>JORDER_E</td>
<td>38.59 / 0.9834</td>
<td>30.50 / 0.8967</td>
<td>32.00 / 0.9347</td>
<td>36.69 / 0.9621</td>
</tr>
<tr>
<td>SIRR [41]</td>
<td>32.37 / 0.9258</td>
<td>22.47 / 0.7164</td>
<td>28.44 / 0.8893</td>
<td>34.02 / 0.9347</td>
</tr>
<tr>
<td>RCDNet</td>
<td>40.00 / 0.9860</td>
<td>31.28 / 0.9093</td>
<td>33.04 / 0.9472</td>
<td>37.71 / 0.9649</td>
</tr>
</tbody>
</table>

5.3. Experiments on synthetic data

Comparison methods and datasets. We then compare our network with the current SOTA single image derain-
ers, including model-based DSC [51], GMM [28], and J-CAS [13]; DL-based Clear [8], DDN [9], RESCAN [27], PReNet [35], SPANet [41], JORDER_E [49], and SIRR [41]; based on four benchmark datasets, including Rain100L, Rain100H [49], Rain1400 [9], and Rain12 [28].

We then analyze the performance of all methods on two real datasets from [41]: the first one (called SPA-Data) contains 638492 rainy/clean image pairs for training and 1000 testing ones, and the second one (called Internet-Data) includes 147 rainy images without groundtruth. Table 3 and Fig. 6 compare the derained results on SPA-Data of all competing methods visually and quantitatively. It is easy to see that even for such complex rain patterns, the proposed RCDNet still achieves an evident superior perfor-

6The code/project links for these comparison methods are listed in supplementary material.
Table 3. PSNR and SSIM comparisons on SPA-Data [41].

<table>
<thead>
<tr>
<th>Methods</th>
<th>Input</th>
<th>DSC</th>
<th>GMM</th>
<th>JCAS</th>
<th>Clear</th>
<th>DDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>34.15</td>
<td>34.95</td>
<td>34.30</td>
<td>34.95</td>
<td>34.39</td>
<td>36.16</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.9269</td>
<td>0.9416</td>
<td>0.9428</td>
<td>0.9453</td>
<td>0.9509</td>
<td>0.9463</td>
</tr>
</tbody>
</table>

Figure 6. Rain removal performance comparisons on a rainy image from SPA-Data. The images are better to be zoomed in on screen.

Figure 7. Derained results for two samples with various rain patterns from Internet-Data. The images are better to be zoomed in on screen.

6. Conclusion

In this paper, we have explored the intrinsic prior structure of rain streaks that can be explicitly expressed as convolutional dictionary learning model, and proposed a novel interpretable network architecture for single image de-raining. Each module in the network can one-to-one correspond to the implementation operators of the algorithm designed for solving the model, and thus the network is almost “white-box” with easily visualized interpretation for all its module elements. Comprehensive experiments implemented on synthetic and real rainy images validate that such interpretability brings a good effect of the proposed network, and especially facilitates the analysis for how it happens in the network and why it works in testing prediction process. The extracted elements through the end-to-end learning by the network, like the rain kernels, are also potentially useful for the related tasks on rainy images.

Acknowledgment. This research was supported by the China NSFC projects under contract 11690011, 61721002, U1811461 and MoE-CMCC “Artifical Intelligence” Project with No. MCM20190701.
References


[45] Wei Wei, Lixuan Yi, Qi Xie, Qian Zhao, Deyu Meng, and Zongben Xu. Should we encode rain streaks in video as deterministic or stochastic? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2516–2525, 2017. 3


