Single Image Deraining Using Bilateral Recurrent Network
Dongwei Ren, Wei Shang, Pengfei Zhu, Qinghua Hu, Deyu Meng and Wangmeng Zuo

Abstract—Single image deraining has received considerable progress based on deep convolutional neural network. Most existing deep deraining methods follow residual learning in image denoising to learn rain streak layer, and perform limited in restoring background image layer. In this work, we first propose a single recurrent network (SRN), where the composition pattern of two layers can be exploited and propagated in multiple stages via LSTM, thereby contributing to the remedy of possible over-removal of rain streaks. This simple SRN is effective not only in learning residual mapping for extracting rain streaks, but also in learning direct mapping for predicting clean background image. Furthermore, we propose bilateral recurrent network (BRN) to allow the interplay between rain streak and background image layers. In particular, two SRNs are coupled to simultaneously exploit these two layers. Instead of naive combination, we propose bilateral LSTMs, which not only can respectively propagate deep features across stages, but also bring the interplay between these two SRNs, which is essential in separating two layers from rainy observation. The experimental results demonstrate that our BRN notably outperforms state-of-the-art deep deraining networks on synthetic datasets quantitatively and qualitatively. The proposed methods also perform more favorably in terms of generalization performance on real-world rainy dataset. All the source code and pre-trained models are available at https://github.com/csdwren/RecDerain.

Index Terms—Image deraining, convolutional neural network, recurrent network, LSTM.

I. INTRODUCTION

Removing rain streaks plays an important role in many computer vision applications in rainy outdoor scenes, e.g., surveillance, object detection and recognition [1], [2]. Single image deraining is a very challenging ill-posed problem, and has received considerable research attention in recent years [3]–[9]. Basically, image deraining can be treated as an image decomposition problem, i.e., a rainy image $y$ should be decomposed into a rain streak layer $r$ and a clean background image layer $x$. There are several conventional optimization based deraining methods [4], [10]–[12] by studying the composition pattern of rainy image and designing proper regularization priors.

With the great success of deep learning in low level vision tasks, e.g., denoising [13]–[17], super-resolution [18]–[21], motion deblurring [22], [23] etc, deep convolutional neural network (CNN)-based deraining methods also achieve significant performance improvements against conventional optimization based methods. By assuming the linear degradation model $y = x + r$, most existing deep deraining networks follow residual learning in image denoising [13] to learn rain streak layer. As seminal deep deraining approaches, Fu et al. decompose a rainy image into a base layer and a detail layer, and utilize a 3-layer CNN [3] or a deeper residual network (ResNet) [24] to extract rain streaks from the detail layer. Yang et al. [7] propose a joint rain streak detection and removal framework by using a dilated convolution network. Furthermore, many network architectures, e.g., residual-guide fusion network [6], rain density aware multi-stream dense network [25], squeeze-and-excitation context aggregation network [5], Laplacian pyramid network [1] and spatial attentive network [26] have been proposed to better extract rain streaks. Most recently, unsupervised learning [27] and semi-supervised

Fig. 1. Deraining results by RESCAN [5], SRN ($T = 6$) at stage $t = 1, 2, 6$, and BRN ($T = 6$), respectively. The result by RESCAN [5] suffers from visible artifacts along with orientations of rain streaks. For SRN, rain streaks can be gradually removed in multiple stages, but the final deraining image still has bright artifacts. The result by BRN is more visually favorable.

Rainy image
RESCAN [5]
SRN ($t = 1$)
SRN ($t = 2$)
SRN ($t = 6$)
BRN

learning [28] have also been studied for training deraining networks.

Despite these various network architectures and training strategies, image deraining actually is a layer decomposition task, where both rain streak layer \( r \) and background image layer \( x \) convey rich structural information. For most existing methods, rain streak layer \( r \) is often over-estimated by deep CNN (see Fig. 8 for background textures in estimated rain streak layer), and then clean background image \( x \) is directly predicted by subtracting rain streak layer \( r \) from rainy image \( y \). They ignore the composition and interplay of rain streak and background image layers, and perform limited in restoring clean background image layer. Moreover, the composition pattern of rainy image is much more complicated than linear degradation model [11], [26], and thus clean background image is often over-subtracted, yielding dark artifacts along with the orientations of rain streaks or over-smoothing textures in deraining images (see the results in Figs. 1 and 9).

In this paper, we first propose a single recurrent network (SRN), which is effective in learning direct mapping from rainy image to clean background image. In particular, SRN basically employs a simple shallow ResNet that is recursively unfolded several times to benefit from progressive deraining in multiple stages. Furthermore, a recurrent layer, i.e., convolutional Long Short-Term Memory (LSTM) [29], [30], is introduced into SRN to exploit the dependencies of deep features across recursive stages, as shown in Fig. 2. The progressive model is crucial to remove rain streak layer. In each stage, original rainy image is also taken as input, by which the composition pattern of two layers can be exploited and propagated in intermediate stages via LSTM, thereby contributing to the remedy of possible over-removal of rain streaks. From Fig. 1, one can see that rain streaks can be progressively removed by SRN in multiple stages.

Furthermore, for the layer separation task, it is reasonable that both rain streak and background image layers should be exploited. SRN is able to model either rain streak layer or background image layer. Thus, we further present bilateral recurrent network (BRN) which exploits bilateral LSTMs (BLSTMs) to allow the interplay between these two layers. Specifically, we employ a SRN \( F_r \) to extract rain streak layer \( r \), which is then fed to another SRN \( F_x \) to predict clean background image \( x \), as shown in Fig. 3. Both \( F_r \) and \( F_x \) are recursively unfolded several times. Instead of naive combination, we propose BLSTMs that are incorporated into \( F_r \) and \( F_x \), finally forming our BRN. As shown in Figs. 3 and 4, by adopting BLSTMs, not only the deep features of rain streak layer and background image layer can be respectively propagated across multiple stages, but also the interplay between \( F_r \) and \( F_x \) would facilitate deraining performance. From Fig. 1, one can see that the result by BRN is more visually plausible than those by SRN and RESCAN [5].

In order to train the deraining networks, hybrid loss functions with careful trade-off parameter tuning are usually adopted to achieve good performance [1], [5], [7], [24], [26], [28], [32], [35] on several benchmark datasets. Moreover, our BRN can be well generalized to handle real-world rainy dataset [26], and perform favorably in generating visually plausible deraining results. We summarize our contributions from three aspects:

- We propose SRN to tackle image deraining in a progressive manner, where the composition pattern of rain streak and background image layers can be exploited and propagated in intermediate stages via LSTM.
- We propose BRN for modeling the interplay between rain streak and background image layers, which is also essential in further improving deraining performance.
- Experimental results demonstrate that SRN and BRN achieve notable performance gains against state-of-the-art methods on synthetic and real rainy images.

The remainder of this paper is organized as follows. Section II reviews related works including optimization-based and deep network-based image deraining methods. Section III presents the proposed SRN and BRN for image deraining. Section IV gives the comparison with state-of-the-art methods, and Section V ends this paper with conclusions and discussions.

## II. Related Work

In this section, we present a brief review on traditional optimization-based and deep network-based image deraining methods.

### A. Optimization-based Deraining Methods

Generally, a rainy image can be represented as the composition of a clean background image layer and a rain streak layer. On the one hand, a linear summation is usually adopted [4], [10], [34], based on which proper regularizers are designed to well pose the deraining problem that is then solved by optimization algorithm. In [10], smoothness regularization and low rank prior are respectively imposed on background image layer and rain streak layer. In [4], two GMMs that are respectively trained on clean image patches and rain streak patches are deployed to model background image layer and rain streak layer. The additive degradation model has also been widely adopted in optimization-based video deraining methods [35]–[39]. On the other hand, it has been suggested that the screen blend model [11] is more realistic for the composition of rainy image. And then the discriminative dictionary learning-based deraining method is proposed [11], where rain streak layer and background image layer are forced to share the fewest dictionary items. However, the composition of real rainy image is much more complicated than existing models, making these optimization-based methods are still facing limited deraining quality.

### B. Deep CNN-based Deraining Methods

A natural deraining solution based on deep learning is to adopt CNNs for directly recovering clean background image
from rainy image. However, the pioneer works [3], [24] suggest that deep CNNs are ineffective in learning direct mapping from rainy image to background image. Instead, Fu et al. propose to remove rain streaks from the detail layer of a rainy image, where a shallow CNN [3] and a deeper ResNet [24] are successively employed. In [7], a very complicated CNN architecture is designed for jointly detecting and removing rain streaks, in which multi-scale dilation filters are adopted to benefit from larger size receptive field. Li et al. [5] propose to recurrently utilize dilated CNN to handle heavy rain streak accumulation. Zhang et al. [25] propose a density aware multi-stream densely connected CNN for joint estimating rain density and removing rain streaks. Besides, there are several works to incorporate lightweight networks in a cascaded scheme [6] or in a Laplacian pyramid framework [1]. Moreover, in [40], the authors propose to take advantage of adversarial learning to enhance the texture details in derained images. Most recently, unsupervised learning [27] and semi-supervised learning [28] are adopted to train deraining networks. To sum up, despite of various network architectures and training strategies, these deep deraining networks learn residual mapping for extracting rain streaks, which are then directly subtracted from rainy image, often yielding over-subtracted dark artifacts especially for heavy rainy images. In this work, we show that our proposed single recurrent network eases the difficulty of learning, and it is feasible to train SRN for learning either direct mapping or residual mapping. Furthermore, our BRN can simultaneously exploit rain streak layer and clean background image layer, leading to more favorable deraining results.

III. PROPOSED METHOD

In this section, we first propose a single recurrent network for direct mapping, which is also effective in residual mapping. Then bilateral recurrent network is proposed by taking advantages of SRN and bilateral LSTMs. Finally, we show that our SRN and BRN models can both be easily trained using a single loss function.

A. Single Recurrent Network

As suggested in [3], [24], a plain deep CNN cannot succeed in directly predicting clean background image from rainy image, and thus deeper and more complicated networks are developed to extract rain streaks via learning residual mapping from rainy image to rain streak layer. However, we suggest that the difficulty of learning deraining network can be eased in multiple stages, and thus it is feasible to train a simple network \( \mathcal{F} \) to learn direct mapping from rainy image to clean background image layer. By unfolding \( \mathcal{F} \) with \( T \) times, the inference at stage \( t \) can be formulated as,

\[
x^t = \mathcal{F}(y, x^{t-1}),
\]

where the parameters of \( \mathcal{F} \) are reused across different stages. It is expected that \( \mathcal{F} \) can remove rain streaks gradually in each stage.

As shown in Fig. 2, we study \( \mathcal{F} \) with four parts: (i) The input layer \( f_{in} \) takes the concatenation of rainy image \( y \) and \( x^{t-1} \) from stage \( t - 1 \) as input, (ii) recurrent layer \( f_{recurrent} \) for propagating dependencies across stages to facilitate progressively removing rain streaks, (iii) several residual blocks (ResBlocks) \( f_{res} \) for extracting deep features, and (iv) output layer \( f_{out} \) for generating deraining background image. This simple network is dubbed single recurrent network (SRN) for image deraining. The inference of SRN \( \mathcal{F} \) at stage \( t \) can be formulated as,

\[
z^t = f_{in}(x^{t-1}, y),
\]

\[
h^t = f_{recurrent}(h^{t-1}, z^t),
\]

\[
x^t = f_{out}(f_{res}(h^t)),
\]

where \( f_{in}, f_{res} \) and \( f_{out} \) are stage-invariant, i.e., network parameters are reused across different stages. The deraining results by SRN in Fig. 1 show that heavy rain streaks accumulation can be gradually removed stage-by-stage. In the following, we present the implementation details of key modules in SRN. We note that all the filters are with size \( 3 \times 3 \), padding \( 1 \times 1 \) and stride 1.

1) Input Layer \( f_{in} \): Generally, \( f_{in} \) is a 1-layer convolution with ReLU nonlinearity \([41]\) for generating deep features \( z^t = f_{in}(x^{t-1}, y) \). Due to the concatenation of 3-channel RGB \( y \) and 3-channel RGB \( x^{t-1} \), the convolution in \( f_{in} \) has 6 channels for input, and the output channel number is 32.

2) Recurrent Layer \( f_{recurrent} \): At stage \( t \), the recurrent layer \( f_{recurrent} \) receives both the features \( z^t \) from input layer and recurrent state \( h^{t-1} \) from stage \( t - 1 \). \( f_{recurrent} \) can be implemented using either convolutional LSTM [29], [30] or convolutional Gated Recurrent Unit (GRU) [42]. In SRN, we choose LSTM since it performs better quantitatively in experiments for image deraining. The LSTM includes an input...
Fig. 3. The architectures of BRN at stage \( t \). BRN consists of two coupled SRNs, i.e., \( F_r \) for extracting rain streaks and \( F_s \) for predicting background image. Instead of two individual LSTMs, BRN can benefit from the interplay between \( F_r \) and \( F_s \) using BLSTMs.

gate \( i^t \), a forget gate \( f^t \), an output gate \( o^t \) and a cell state \( c^t \), and can be formulated as,

\[
\begin{align*}
i^t &= \sigma(W_{iz} \otimes z^t + W_{ih} \otimes h^{t-1} + b_i), \\
f^t &= \sigma(W_{fz} \otimes z^t + W_{fh} \otimes h^{t-1} + b_f), \\
o^t &= \sigma(W_{oz} \otimes z^t + W_{oh} \otimes h^{t-1} + b_o), \\
g^t &= \tanh(W_{gz} \otimes z^t + W_{gh} \otimes h^{t-1} + b_g), \\
c^t &= f^t \odot c^{t-1} + i^t \odot g^t, \\
h^t &= o^t \odot \tanh(c^t),
\end{align*}
\]

where \( \odot \) is 2D convolution, \( \circ \) is entry-wise product, \( \sigma \) is sigmoid function, \( W \) and \( b \) are corresponding convolution matrix and bias vector. All the convolutions in LSTM have 32 input channels and 32 output channels.

3) Residual Blocks \( f_{res} \): \( f_{res} \) is the key component to extract deep representation for deraining. We implement \( f_{res} \) with 5 ResBlocks as its simplest form, i.e., each ResBlock includes 2 convolution layers followed by ReLU [41]. All the convolution layers receive 32-channel features without downsampling or upsampling operations.

4) Output Layer \( f_{out} \): \( f_{out} \) takes deep features from \( f_{res} \) with 32 channels as input, and generates deraining RGB image with 3 channels.

B. Learning Residual Mapping via SRN

Analogously, SRN can be adopted to learn residual mapping for extracting rain streaks, which can be formulated as,

\[
\begin{align*}
r^t &= F_r(y, r^{t-1}), \\
x^t &= y - r^t,
\end{align*}
\]

where \( F \) shares the same architecture as that in Eqn. (1), except substituting rain streak layer \( r \) as input and output. Due to its narrower mapping range, SRN with residual mapping can achieve a little better results than SRN with direct mapping (refer to the comparison in Table II). Nevertheless, our SRN is more effective in both extracting rain streaks and predicting background image against existing complicated deraining networks.

C. Bilateral Recurrent Network

Although SRN with residual mapping can achieve better quantitative results, its extracted rain streak layer contains some textures from background image (see Fig. 8), and thus the over-subtracted deraining image is likely to be over-smoothed. Taking one step further, we propose to stack two SRNs for extracting rain streaks and predicting clean background image, respectively. At stage \( t \), it can be formulated as,

\[
\begin{align*}
r^t &= F_r(y, r^{t-1}), \\
x^t &= F_s(x, x^{t-1}, r^t),
\end{align*}
\]

where \( F_r \) and \( F_s \) are two coupled SRNs for extracting rain streak layer and clean background image layer, respectively, resulting in coupled recurrent network (CRN) for image deraining. In CRN, two individual LSTMs are adopted to propagate deep features across rain streak layer and background image layer, respectively. We in this work suggest that the interplay between these two layers would facilitate deraining performance. To this end, we first propose bilateral LSTMs (BLSTMs).

1) Bilateral LSTMs: Besides propagating the hidden states across stages, we propose to bring the interplay between these two LSTMs, forming BLSTMs. As shown in Fig. 4, the hidden state \( h_r \) in layer \( r \) is not only propagated across stages to facilitate rain streak extraction, but also is fed to the BLSTM in layer \( x \), and vice versa.

In particular, BLSTM in \( F_r \) at stage \( t \) receives the features \( z^t = f_r(y, r^{t-1}) \) from input layer \( f_r(\cdot) \), recurrent state \( h_r^{t-1} \) in rain streak layer and recurrent state \( h_x^{t-1} \) from clean background image layer as the input. And then both \( F_r \) and \( F_s \) can be formally expressed as,

\[
\begin{align*}
i_r^t &= \sigma(W_{riz} \otimes z^t + W_{rizh} \otimes h^{t-1} + W_{rizb} + b_i), \\
f_r^t &= \sigma(W_{rfz} \otimes z^t + W_{rfzh} \otimes h^{t-1} + W_{rfzb} + b_f), \\
o_r^t &= \sigma(W_{roz} \otimes z^t + W_{rozh} \otimes h^{t-1} + W_{rozb} + b_o), \\
g_r^t &= \tanh(W_{rgz} \otimes z^t + W_{rgzh} \otimes h^{t-1} + W_{rgzb} + b_g), \\
c_r^t &= f_r^t \odot c_r^{t-1} + i_r^t \odot g_r^t, \\
h_r^t &= o_r^t \odot \tanh(c_r^t),
\end{align*}
\]

BLSTM in \( F_x \) has almost the same architecture with that in \( F_r \) by substituting subscripts \( r \) and \( x \). And the minor distinctions are two-fold: (i) \( z^t = f_x(y, x^{t-1}, r^t) \) is received as input, and (ii) recurrent state \( h_x^{t-1} \) instead of \( h_r^{t-1} \) is taken into account. And the BLSTM in \( F_x \) is formulated as,

\[
\begin{align*}
i_x^t &= \sigma(W_{xiz} \otimes z^t + W_{xizh} \otimes h^{t-1} + W_{xizb} + b_i), \\
f_x^t &= \sigma(W_{xfz} \otimes z^t + W_{xfzh} \otimes h^{t-1} + W_{xfzb} + b_f), \\
o_x^t &= \sigma(W_{rox} \otimes z^t + W_{rozh} \otimes h^{t-1} + W_{roxb} + b_o), \\
g_x^t &= \tanh(W_{xgz} \otimes z^t + W_{xgh} \otimes h^{t-1} + W_{xgzb} + b_g), \\
c_x^t &= f_x^t \odot c_x^{t-1} + i_x^t \odot g_x^t, \\
h_x^t &= o_x^t \odot \tanh(c_x^t).
\end{align*}
\]

By adopting BLSTMs, deep features can be propagated between two layers and across multiple stages.

2) Bilateral Recurrent Network: As shown in Fig. 3, two SRNs \( F_r \) and \( F_s \) can be recursively unfolded \( T \) times. BLSTM is incorporated into \( F_r \) and \( F_s \) to respectively propagate deep features of \( F_r \) and \( F_s \) across stages and bring the interplay between \( F_r \) and \( F_s \), forming BRN. At stage \( t \), BRN can be formulated as,

\[
\begin{align*}
r^t &= F_r(y, r^{t-1}, (h_r^{t-1}, h_x^{t-1})), \\
x^t &= F_s(x, x^{t-1}, r^t, (h_x^{t-1}, h_h^{t-1})),
\end{align*}
\]
where $h_r$ and $h_c$ are recurrent states for rain streak layer and background image layer in BLSTMs, respectively.

Generally, $F_r$ and $F_x$ share the similar SRN architecture. $F_r$ includes 1 input layer, BLSTM, 3 ResBlocks and 1 output layer. The input layer includes 1 convolution layer and ReLU, and receive the concatenation of rainy image $y$ and previous rain streak layer $r^{t-1}$ as the input. The only distinctions of $F_x$ with $F_r$ are two-fold: (i) 5 ResBlocks are adopted in $F_x$, since the background images are usually with richer structures and textures. (ii) The input layer receives not only the concatenation of the rainy image $y$ and the estimated background image $x^{t-1}$, but also the current rain streak layer $r^t$. By simultaneously considering rain streak layer and background image layer, the composition pattern of rainy images can be implicitly learned by $F_x$. The other parameters including convolution channels, kernel size, padding and stride have the same settings with SRN.

### D. Training SRN and BRN

As for training SRN with $T$ stages, we have $T$ outputs, i.e., $x^1$, $x^2$, ..., $x^T$, and it is natural to impose recursive supervision on every intermediate result,

$$
\mathcal{L} = \sum_{t=1}^{T} \lambda_t \ell (x^t, x^{gt}) ,
$$

where $x^{gt}$ is the corresponding ground-truth clean image, $\ell (\cdot , \cdot)$ measures the difference between the output of SRN at stage $t$ and the corresponding ground-truth, and $\lambda_t$ is a trade-off parameter.

For BRN with $T$ stages, we have $T$ estimated background images and $T$ rain streak layers, i.e., $x^1$, $x^2$, ..., $x^T$ and $r^1$, $r^2$, ..., $r^T$. Similar to Eqn (9), the recursive supervision on background image layer and rain streak layer can be defined as,

$$
\mathcal{L}_x = \sum_{t=1}^{T} \lambda_t \ell (x^t, x^{gt}) ,
$$

$$
\mathcal{L}_r = \sum_{t=1}^{T} \lambda_t \ell (x^t, x^{gt}) .
$$

Then $\mathcal{L}_r$ and $\mathcal{L}_x$ should be balanced by two trade-off parameters $\alpha$ and $\beta$,

$$
\mathcal{L} = \alpha \mathcal{L}_r + \beta \mathcal{L}_x .
$$

Moreover, it is worth noting that we have tried to train SRN and BRN by only imposing supervision on the final output $x^T$,

$$
\mathcal{L} = \ell (x^T, x^{gt}) .
$$

It is interesting to see that the single loss function can lead to better performance for both SRN and BRN (see the results in Sec. IV-A1).

As for the choice of $\ell (\cdot , \cdot)$, several hybrid loss functions, e.g., $\text{MSE+SSIM}$ [6], $\ell_1+\text{SSIM}$ [1] or adversarial loss [40], have been widely adopted for training deraining networks. Nevertheless, we empirically find that a single loss function, e.g., $\text{MSE}$ loss or negative $\text{SSIM}$ loss [31], is sufficient to train both SRN and BRN, which can be attributed that our proposed recurrent network architecture eases the difficulty of learning. Given an image $a$ and its corresponding ground-truth image $a^{gt}$, the negative $\text{SSIM}$ loss and $\text{MSE}$ loss can be respectively formulated as,

$$
\ell (a, a^{gt}) = -\text{SSIM} (a, a^{gt}) ,
$$

$$
\ell (a, a^{gt}) = \| a - a^{gt} \|^2 .
$$

And the experimental results in Sec. IV-A1 indicate that a single negative $\text{SSIM}$ loss can achieve superior results quantitatively and qualitatively.

### IV. EXPERIMENTAL RESULTS

In this section, we first conduct ablation studies to verify the key contributions of SRN and BRN, then evaluate SRN and BRN against state-of-the-art deraining methods on several benchmark datasets quantitatively and qualitatively, and finally validate their generalization performance on real-world rainy images. More results can be found at https://github.com/csdwren/RecDerain.

Our SRN and BRN are implemented using Pytorch [43], and are trained on a PC equipped with two NVIDIA GTX 1080Ti GPUs. In our experiments, SRN and BRN share the same training settings except specific declarations. The patch size is $100 \times 100$. The batch size is 18 for SRN and is 12 for BRN. The ADAM [44] algorithm is adopted to train the models with an initial learning rate $1 \times 10^{-3}$, and ends after 100 epochs. When reaching 30, 50 and 80 epochs, the learning rate is decayed by multiplying 0.2.

#### A. Ablation Study

All the ablation studies are conducted on a heavy rainy dataset [7] with 1,800 rainy images for training and 100 rainy images (Rain100H) for testing. The training and testing images for Rain100H have been updated recently on their project website. However, the reported results in recent deraining works are still based on the original dataset. To keep the consistent comparison, we thus adopt the original dataset for evaluation, where we strictly exclude 546 rainy images from the 1,800 training samples since they have the same background contents with testing images. The performance is quantitatively evaluated using two popular metrics in low-level vision tasks, i.e., $\text{SSIM}$ [31] and PSNR.

For both SRN and BRN, the trade-off parameters $\{\lambda_t\}_{t=1}^{T}$ are set as $\lambda_t = 0.5 \ (t = 1, 2, \ldots, T - 1)$ and $\lambda_T = 1.5$, where the trade-off parameter for the final stage is larger than the others. For BRN, the trade-off parameters $\alpha = 0.45$ and $\beta = 0.55$ are adopted. Moreover, the stage number $T = 6$ is set for SRN and BRN except in the subsection for discussing stage number selection.
1) Loss Functions: We discuss the effects of single loss and recursive loss, and the comparison of single MSE loss and single negative SSIM loss.

**Single Loss vs. Recursive Loss.** By taking negative SSIM loss Eqn. (14), SRN and BRN are trained using single SSIM loss (SRN-SSIM and BRN-SSIM) and recursive SSIM loss (SRN-RecSSIM and BRN-RecSSIM), respectively. From Table I, it is interesting to see that SRN-RecSSIM and BRN-RecSSIM perform moderately inferior to SRN-SSIM and BRN-SSIM, respectively. The results indicate that a single loss on the final stage is sufficient to train our SRN and BRN.

**MSE Loss vs. Negative SSIM Loss.** Then we discuss the performance of negative SSIM loss against MSE loss (SRN-MSE and BRN-MSE) under the single supervision. From Table I, one can see that SRN-SSIM and BRN-SSIM are better in terms of not only SSIM metrics but also PSNR metrics. We empirically suggest that a single negative SSIM loss is sufficient to train our SRN and BRN models.

### TABLE I

<table>
<thead>
<tr>
<th></th>
<th>SRN-MSE</th>
<th>SRN-SSIM</th>
<th>SRN-RecSSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>29.08</td>
<td>29.32</td>
<td>29.12</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.880</td>
<td>0.898</td>
<td>0.895</td>
</tr>
</tbody>
</table>

(a) Comparison of SRN models with different loss functions.

<table>
<thead>
<tr>
<th></th>
<th>BRN-MSE</th>
<th>BRN-SSIM</th>
<th>BRN-RecSSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>29.40</td>
<td>30.47</td>
<td>30.31</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.885</td>
<td>0.913</td>
<td>0.908</td>
</tr>
</tbody>
</table>

(b) Comparison of BRN models with different loss functions.

2) Network Architecture of SRN: In this subsection, we assess the contributions of several key modules of SRN, including recurrent layer, network input, and provide the comparison of residual mapping and direct mapping.

**Recurrent Layer.** In SRN, we test two types of recurrent layers, i.e., LSTM (SRN-LSTM) and GRU (SRN-GRU). It can be seen from Table II that LSTM performs slightly better than GRU in terms of quantitative metrics, and thus is adopted as the default implementation of recurrent layer in our methods.

### TABLE II

<table>
<thead>
<tr>
<th></th>
<th>SRN&lt;sub&gt;x&lt;/sub&gt;</th>
<th>SRN-LSTM</th>
<th>SRN-GRU</th>
<th>SRN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>28.91</td>
<td>29.32</td>
<td>29.08</td>
<td>29.46</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.895</td>
<td>0.898</td>
<td>0.896</td>
<td>0.899</td>
</tr>
</tbody>
</table>

Network Input. We test a variant of SRN by only taking $x_{t-1}$ at each stage as input to $f_{lin}$ (i.e., SRN<sub>x</sub>), where such strategy has been adopted in [5], [7]. From Table II, SRN<sub>x</sub> is obviously inferior to SRN in terms of both PSNR and SSIM, indicating the benefit of receiving $y$ at each stage.

**Direct Mapping vs. Residual Mapping.** By adopting the same SRN architecture, we use SRN to learn direct mapping (i.e., SRN-LSTM in Table II) for predicting background image and to learn residual mapping (i.e., SRN in Table II) for extracting rain streaks. From Table II, SRN is effective in directly generating clean background image, and residual mapping can make a further contribution to performance gain. To sum up, benefited from the recurrent networks, it is feasible to learn SRN in either residual mapping or direct mapping for image deraining, both leading to appealing deraining performance.

3) Network Architecture of BRN: Based on the observations in SRN, we adopt LSTM and BLSTMs in CRN and BRN. To validate the effectiveness of BLSTMs in BRN, we provide several variants of BRN, including CRN (with two individual BLSTMs) and two BLSTMs variants, i.e., BRN<sub>r→x</sub> (information only from $F_r$ to $F_x$) and BRN<sub>x→r</sub> (information only from $F_x$ to $F_r$). From Table III, one can see that BRN can achieve higher average PSNR and SSIM metrics than CRN, BRN<sub>r→x</sub> and BRN<sub>x→r</sub>. It is worth noting that BRN<sub>r→x</sub> is inferior to CRN, and it is owing to that we only supervise the final stage of $x$, possibly leading to inaccurate rain streaks $r$. And thus the monodirectional information from $F_r$ to $F_x$ would damage the performance. But the interplay between $F_r$ and $F_x$ can make significant improvements against all the three variants, indicating the effectiveness of BLSTMs. We also report the visual quality comparison of these variants in Fig.5.

4) Variants of SRN and BRN: We present the comparison of variants of SRN and BRN.

**Bilateral GRUs vs. Bilateral LSTMs.** By adopting the same training settings with BRN, a variant BRN-GRU is trained...
to verify the effectiveness of bilateral GRUs. In BRN-GRU, LSTMs in both $F_r$ and $F_x$ are substituted as GRUs [42]. On Rain100H dataset [7], BRN with bilateral LSTMs outperforms BRN-GRU in terms of both average PSNR and SSIM values, as reported in Table IV. The PSNR decrease $\sim 0.4$dB by BRN-GRU is notable, which can also be verified in terms of visual quality as shown in Fig. 7. Therefore, bilateral LSTMs is a better choice against bilateral GRU for BRN.

**Bilateral Recurrent Architecture vs. Single Recurrent Architecture.** One may argue that SRN with more parameters would perform better than BRN. As reported in Table IV, by increasing ResBlocks number to 18, SRN$^+$ has the larger number of parameters than BRN and BRN-GRU. SRN$^+$ indeed achieves better performance than SRN, but it is significantly inferior to BRN. It is worth noting that BRN-GRU has 20% less amount of parameters than SRN$^+$, but it still performs better than SRN$^+$ in terms of both PSNR and SSIM values. Thus, the superiority of BRN should be mainly ascribed to the bilateral recurrent architecture instead of simply increasing the number of parameters.

### TABLE III
**Effectiveness of BLSTMs in BRN.**

<table>
<thead>
<tr>
<th></th>
<th>CRN</th>
<th>$BRN_{x\rightarrow r}$</th>
<th>$BRN_{r\rightarrow x}$</th>
<th>BRN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PSNR</strong></td>
<td>30.04</td>
<td>30.22</td>
<td>30.01</td>
<td>30.47</td>
</tr>
<tr>
<td><strong>SSIM</strong></td>
<td>0.909</td>
<td>0.910</td>
<td>0.908</td>
<td>0.913</td>
</tr>
</tbody>
</table>

### TABLE IV
**Comparison of variants of SRN and BRN on Rain100H [7].** SRN$^+$ is a variant of SRN by increasing ResBlocks number to 18. BRN-GRU is a variant of BRN by substituting LSTMs with GRUs.

<table>
<thead>
<tr>
<th></th>
<th>SRN</th>
<th>SRN$^+$</th>
<th>BRN-GRU</th>
<th>BRN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PSNR</strong></td>
<td>29.32</td>
<td>29.99</td>
<td>30.08</td>
<td>30.47</td>
</tr>
<tr>
<td><strong>SSIM</strong></td>
<td>0.898</td>
<td>0.906</td>
<td>0.908</td>
<td>0.913</td>
</tr>
<tr>
<td><strong>#Parameters</strong></td>
<td>168,963</td>
<td>409,411</td>
<td>320,166</td>
<td>375,526</td>
</tr>
</tbody>
</table>

5) **Recurrent Stage Number T**: We hereby discuss the stage number $T$ for SRN and BRN. Fig. 6 demonstrates PSNR and SSIM values of SRN and BRN models by setting recurrent stage number $T = 1, 2, 3, \ldots, 9$ trained using single negative SSIM loss. One can see that SRN and BRN have the consistent performance increases along with increasing stage number $T$ from 1 to 6. When $T > 6$, we can observe the performance drops or perturbations for SRN and BRN. Especially, $T = 9$ yields performance drops for both SRN and BRN, possibly due to too long backward propagation distance. Thus, in the following experiments, we set $T = 6$ for both SRN and BRN in comparison with state-of-the-art algorithms.

**B. Comparison with State-of-the-arts on Synthetic Datasets**

We first evaluate BRN on three benchmark datasets, i.e., Rain100H [7], Rain100L [7] and Rain12 [4], and compare it with several state-of-the-art deraining methods including conventional optimization-based method: GMM [4], state-of-the-art supervised deep CNN-based methods: DDN [24], ResGuideNet [6], JORDER [7] and RESCAN [5], and semi-supervised deep deraining method: SIRR [28]. For heavy rainy images (Rain100H) and light rainy images (Rain100L), the models are respectively trained, and the models for light rain are used to process Rain12. Since the source codes of ResGuideNet are not available, we borrow the quantitative results from [6]. As for JORDER, we directly calculate average PSNR and SSIM on deraining results provided by the authors. As for semi-supervised SIRR [28], we adopt their pre-trained model where SIRR model is fine-tuned using real-world rainy images. It is worth noting that RESCAN and BRN are trained on strict 1,254 rainy images for Rain100H. The comparison results are reported in Table V, from which we can see that our BRN is notably superior to the existing methods. Especially for Rain100H with heavy rain streaks, the performance gain by our BRN is very significant. The semi-supervised SIRR is moderately inferior to the other supervised deraining methods, because real-world rainy images are quite different from these synthetic datasets. As shown in Fig.8, BRN can effectively extract rain streak layer $r$ using $F_r$, while the rain streak layers estimated by RESCAN has background textures.

Moreover, the authors [33] have updated the datasets of Rain100H and Rain100L, both of which include 1,800 training pairs and 200 testing images. We note them as RainHeavy* and RainLight*, respectively. And we take more recent state-of-the-art methods into comparison on RainHeavy* and RainLight*, including DDN [24], SIRR [28], SPA-Net [26], RESCAN [5], PReNet [32] and JORDER-E [33]. Due to the inconsistent evaluation metrics adopted in their original papers, we re-train all these models for RainHeavy* and RainLight*. Average PSNR and SSIM values are reported in Table VII. One can see that our BRN achieves much higher quantitative...
Fig. 7. Visual comparison of variants of SRN and BRN. SRN+ is a variant of SRN by increasing ResBlocks number to 18. BRN-GRU is a variant of BRN by substituting LSTMs with GRUs.

TABLE V
AVERAGE PSNR AND SSIM COMPARISON ON SYNTHETIC DATASETS, INCLUDING RAIN100H [7], RAIN100L [7] AND RAIN12 [4].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rain100H</td>
<td>15.05/0.425</td>
<td>21.92/0.764</td>
<td>25.25/0.841</td>
<td>26.54/0.835</td>
<td>28.64/0.864</td>
<td>22.47/0.716</td>
<td>30.47/0.913</td>
</tr>
<tr>
<td>Rain100L</td>
<td>28.66/0.865</td>
<td>32.16/0.936</td>
<td>33.16/0.963</td>
<td>36.61/0.974</td>
<td>—</td>
<td>32.37/0.926</td>
<td>38.16/0.982</td>
</tr>
<tr>
<td>Rain12</td>
<td>32.02/0.855</td>
<td>31.78/0.900</td>
<td>29.45/0.938</td>
<td>33.92/0.953</td>
<td>—</td>
<td>34.02/0.935</td>
<td>36.74/0.959</td>
</tr>
</tbody>
</table>

Fig. 8. Comparison of extracted rain streak layers by BRN and RESCAN [5]. The rain streak layer by RESCAN contains background textures, while the one by BRN is clearer.

TABLE VII
AVERAGE PSNR AND SSIM COMPARISON ON NEW DATASETS OF RAIN100H AND RAIN100L, I.E., RAINHeavy* AND RAINLight* [33].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RainHeavy*</td>
<td>22.03/0.713</td>
<td>28.02/0.862</td>
<td>22.17/0.719</td>
<td>26.59/0.869</td>
<td>29.36/0.903</td>
<td>30.21/0.891</td>
<td>30.27/0.917</td>
</tr>
<tr>
<td>RainLight*</td>
<td>31.66/0.922</td>
<td>38.43/0.982</td>
<td>32.20/0.929</td>
<td>36.13/0.975</td>
<td>37.93/0.983</td>
<td>39.13/0.985</td>
<td>38.86/0.985</td>
</tr>
</tbody>
</table>

TABLE VIII
GENERALIZATION EVALUATION ON REAL-WORLD RAINY DATASET SPAData [26], WHERE THE MODELS ARE TRAINED FOR RAINLight* [33].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>34.80</td>
<td>34.73</td>
<td>34.84</td>
<td>35.26</td>
<td>35.00</td>
<td>34.13</td>
<td>35.34</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.936</td>
<td>0.937</td>
<td>0.936</td>
<td>0.945</td>
<td>0.941</td>
<td>0.934</td>
<td>0.945</td>
</tr>
</tbody>
</table>
metrics than all the competing methods on RainHeavy*. On RainLight*, BRN is only a little inferior to JORDER-E in terms of PSNR, but is comparable in terms of SSIM. In Fig. 9, BRN can remove rain streaks clearly.

Then we evaluate BRN on another dataset Rain1400 [24], including 12,600 rainy images for training and 1,400 rainy images for testing. As reported in Table VI, BRN achieves more significant gains than DDN and SIRR. The visual quality improvement is also very significant, while the results by DDN and SIRR still have visible rain streaks in Fig. 10. The performance improvements in quantitative metrics and visual quality can be attributed to the bilateral modeling capacity of BRN.

Finally, the comparison of computational efficiency is conducted. Considering optimization-based method GMM is very time-consuming and the code of ResGuideNet is not released, we report the inference time of DDN [24], RESCAN [5], SIRR [28], SPANet [26], PReNet [32] and JORDER-E [33] for image sizes $512 \times 512$ and $1024 \times 1024$. The running time is recorded on one NVIDIA GTX 1080Ti GPU. As reported in Table IX, BRN is more efficient than the competing methods except our previous PReNet [32]. Meanwhile, considering better performance in Table VII and better generalization ability in Table VIII, our BRN can be a preferred choice for practical applications.

### C. Evaluation on Real Rainy Images

In this subsection, we evaluate the generalization ability of BRN against state-of-the-art competing algorithms on real-world rainy images. The authors in [26] established a real-world rainy dataset, i.e., SPAData, containing 1,000 testing rainy and clean image pairs from rainy videos. Our methods are compared with DDN [24], RESCAN [5], SIRR [28], SPANet [26], PReNet [32] and JORDER-E [33]. We note that the semi-supervised SIRR [28] network is fine-tuned using many real-world rainy images in unsupervised learning, while the other supervised methods are trained for RainLight*. The quantitative metrics are reported in Table VIII, from which one can see that BRN can generalize well to real-world rainy images. From Tables VII and VIII, albeit BRN is slightly inferior to JORDER-E on RainLight*, it achieves much higher metrics (i.e., more than 1dB in PSNR) when applying to real rainy images. As shown in Fig. 11, BRN is able to clearly remove rain streaks, while the other methods suffer from remaining visible rain streaks.

### V. Conclusion

In this paper, we first proposed a simple yet effective single recurrent network (i.e., SRN) for image deraining. SRN is effective not only in learning residual mapping for extracting rain streaks but also in learning direct mapping for predicting background image. Then by coupling two SRNs and
incorporating BLSTMs, BRN was proposed to sequentially extract rain streaks and generate clean background image, where the interplay between rain streak layer and clean background image layer can be exploited. Both SRN and BRN can be easily trained using only a single negative SSIM loss. Extensive experimental results validate the superiority of BRN in deraining performance on several benchmark datasets against state-of-the-art deraining methods. And our trained models can be better generalized to real-world rainy images. Additionally, the ideas of progressive and BLSTMs models may also be beneficial to other layer separation tasks, e.g., reflection removal, shadow removal and fences removal, which will be left for further studies in future work.

REFERENCES


