Deep Convolutional and Recurrent Neural Network for Single Image Deraining

Anonymous ECCV submission

Paper ID 618

Abstract. Rain streaks can severely degrade the visibility, which causes many current computer vision algorithms fail to work. So it is very necessary to remove the rain from images. We propose a novel deep network architecture based on deep convolutional and recurrent neural networks for single image deraining. As contextual information is very important for rain removal, we first adopt the dilated convolutional neural network to acquire large receptive field. To better fit the rain removal task, we also modify the network. In heavy rain, rain streaks have various directions and shapes, which can be regarded as the accumulation of multiple rain streak layers. We assign different weights to various rain streak layers according to their intensities by incorporating the squeeze-and-excitation block. Since rain streak layers overlap with each other, it is not easy to remove the rain in one stage. So we further decompose the rain removal into multiple stages. Recurrent neural network is incorporated to preserve the useful information in previous stages and benefit the rain removal in later stages. We conduct extensive experiments on both synthetic and real-world datasets and our proposed method outperforms the state-of-the-art approaches under all evaluation metrics.

1 Introduction

Rain is a very common weather in actual life. However, it can affect the visibility. Especially in heavy rain, rain streaks from various directions accumulate and make the background scene misty, which will seriously influence the accuracy of many computer vision systems, including video surveillance, object detection and tracking in autonomous driving, and etc. Therefore, it is an important task to remove the rain and recover the background from rain images.

Image deraining has attracted much attention in the past decade. Many methods have been proposed to solve this problem. Existing methods can be divided into two categories, including video based approaches and single image based approaches. As video based methods can utilize the relationship between frames, it is relatively easy to remove rain from videos [1–4]. However, single image deraining is more challenging, and we mainly focus on this task in this paper. For single image deraining, traditional methods, such as discriminative sparse coding [5], low rank representation [6], and the Gaussian mixture model [7], have been applied to this task and they work quite well. Recently, deep learning based deraining methods [8, 9] receive extensive attention due to its powerful ability of feature representation. All these related approaches achieve good performance, but there is still much space to improve.

There are mainly two limitations of existing approaches. On the one hand, according to [10–12], spatial contextual information is very useful for deraining. However, many current methods remove rain streaks based on local image patches, which neglect the contextual information in large regions. On the other hand, as rain steaks in heavy rain have various directions and shapes, they blur the scene in different ways. It is a common way [9, 13] to decompose the overall rain removal problem into multiple stages, so that we can remove rain streaks iteratively. Since these different stages work together to remove rain streaks, the information of deraining in previous stages is useful to guide and benefit the rain removal in later stages. However, existing methods treat these rain streak removal stages independently and do not consider their correlations.

Motivated by the above two issues, we propose a novel deep network for single image deraining. The pipeline of our proposed network is shown in Fig. 1. We remove rain streaks stage by stage. At each stage, we use the context aggregation network with multiple full-convolutional layers to remove rain streaks. As rain streaks have various directions and shapes, each channel in our network corresponds to one kind of rain streak. Squeeze-and-Excitation (SE) blocks are used to assign different weights to various channels according to their interdependencies in each convolution layer. Benefited from the exponentially increased convolution dilations, our network has a large reception field with low depth, which can help us to acquire more contextual information. To better utilize the useful information for rain removal in previous stages, we further incorporate the RNN architecture with three kinds of recurrent units to guide the deraining in later stages. We name the proposed deep network as **RE**current **S**E **C**ontext **A**ggregation **N**et (RESCAN).

Main contributions of this paper are listed as follows:

1. We propose a novel unified deep network for single image deraining. By which, we remove the rain stage by stage. At each stage, we use the contextual dilated network to remove the rain. SE blocks are used to assign different weight to various rain streak layers according to their properties.
2. This is the first paper to consider the correlations between different stages of rain removal. By incorporating RNN architecture with three kinds of recurrent units, the useful information for rain removal in previous stages can be incorporated to guide the deraining in later stages. Our network is suitable for recovering rain images with complex rain streaks, especially in heavy rain.
3. Our deep network achieves superior performance compared with the state-of-the-art methods on various datasets.

2 Related Works

During the past decade, many methods have been proposed to separate rain streaks and background scene from rain images. Existing deraining methods can

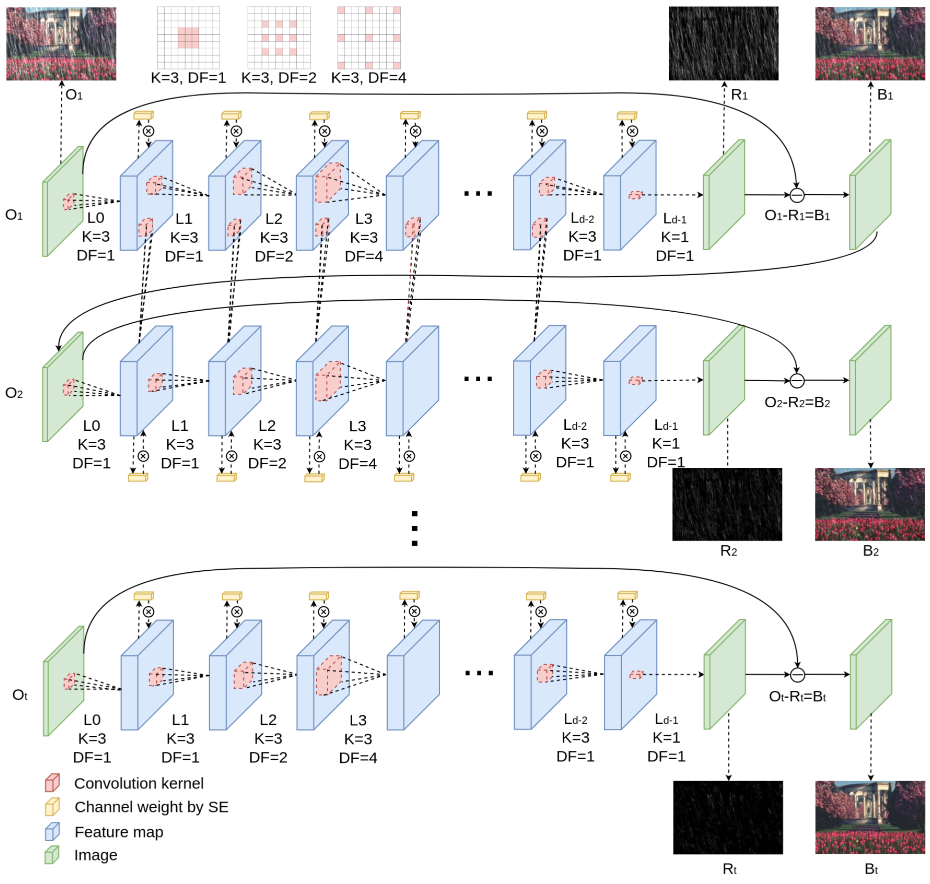

Fig. 1. The unfolded architecture of **RE**current **SE** **C**ontext **A**ggregation **N**etwork(RESCAN). K is the convolution kernel size, and DF indicates the dilation factor.

be categorized into two different types, including video based methods and single image based methods. We briefly review these related methods as follows.

Video base Methods As video based methods can leverage the temporal information by analyzing the difference between adjacent frames, it is relatively easy to remove the rain from videos [14, 3]. Garg and Nayar [15, 16, 2] propose an appearance model to describe rain streaks based on photometric properties and temporal dynamics. Meanwhile, Zhang et al. [1] exploits temporal and chromatic properties of rain in videos. Bossu et al. [17] detects the rain based on the histogram of orientation of rain streaks. In [4], the authors provide a review of video-based deraining methods proposed in recent years.

Single image based methods Compared with video deraining, single image deraining is much more challenging, since there is no temporal information in images. For this task, traditional methods, including dictionary learning [18], Gaussian mixture models (GMMs) [19], and low-rank representation [20], have been widely applied. Based on dictionary learning, Kang et al. [21] decomposes high frequency parts of rain images into rain and nonrain components. Wang et al. [22] defines a 3-layer hierarchical scheme. Luo et al. [5] proposes a discriminative sparse coding framework based on image patches. Gu et al. [23] integrates analysis sparse representation (ASR) and synthesis sparse representation (SSR) to solve a variety of image decomposition problems. In [7], GMM works as a prior to decompose a rain image into background and rain streaks layer. Chang et al. [6] leverages the low-rank property of rain streaks to separate two layers. Zhu et al. [24] combines three different kinds of image priors.

Recently, several deep learning based deraining methods achieve promising performance. Fu et al. [8] first introduces deep learning methods to deraining problem. Similar to [21], they also decompose rain images into low- and high-frequency parts, and then map high-frequency part to rain streaks layer using deep residual network. Yang et al. [9] designs a deep recurrent dilated network to jointly detect and remove rain streaks. Zhang et al. [25] uses the generative adversarial network (GAN) to prevent the degeneration of background image when it is extracted from rain image, and utilizes the perceptual loss to further ensure better visual quality. Li et al. [13] designs a novel multi-stage convolutional neural network that consists of several parallel sub-networks, each of which is made aware of different scales of rain streaks.

3 Rain Models

It is a commonly used rain model to decompose the observed rain image $\mathbf{O}$ into the linear combination of the rain-free background scene $\mathbf{B}$ and the rain streak layer $\mathbf{R}$:

$$\mathbf{O} = \mathbf{B} + \mathbf{R}. \tag{1}$$

By removing rain streaks layer R from the observed image $\mathbf{O}$, we can obtain the rain-free scene B.

Based on the rain model in Eq. 1, many rain removal algorithms assume that rain steaks should be sparse and have similar characters in falling directions and shapes. However, in reality, raindrops in the air have various of appearances, and occur in different distances from the camera, which leads to the irregular distribution of rain streaks. In this case, single rain streak layer $\mathbf{R}$ is not enough to well simulate this complex situation.

To reduce the complexity, we regard rain streaks with similar shape or depth as one layer. Then we can divide the captured rainy scene into the combination of several rain streak layers and a unpolluted background. Based on this, we can reformulate the rain model as follows:

$$\mathbf{O} = \mathbf{B} + \sum_{i=1}^{n} \mathbf{R}^i, \tag{2}$$

where $\mathbf{R}^i$ represents one kind of rain streaks, and n is the total number of different rain streak layers.

According to [26], things in reality might be even worse, especially in heavy rain situation. Accumulation of multiple rain streaks in the air may cause attenuation and scattering, which further increase the diversity of rain streaks' brightness. For camera or eye visualization, the scattering causes haze or frog effects. This further pollutes the observed image $\mathbf{O}$. For camera imaging, due to the limitation of pixels number, rain streaks far away cannot occupy full pixels. When mixed with other things, the image will be blurry. To handle the issues above, we further take the global atmospheric light into consideration and assign different weights to various rain streak layers according to their intensities. We further generalize the rain model to:

$$\mathbf{O} = \left(1 - \sum_{i=1}^{n} \alpha_i\right) \mathbf{B} + \alpha_1 \mathbf{A} + \sum_{i=2}^{n} \alpha_i \mathbf{R}^i, \tag{3}$$

where $\mathbf{A}$ is the global atmospheric light, α_1 is the scene transmission, α_i $(i = 1, \cdots, n)$ indicates the brightness of a rain streak layer or a haze layer.

4 Deraining Method

Instead of using decomposition methods with artificial priors to solve the problem in Eq. 3, we intend to learn a function f that directly maps observed rain image $\mathbf{O}$ to rain streak layer $\mathbf{R}$, since $\mathbf{R}$ is sparser than $\mathbf{B}$ and has simpler texture. Then we can subtract $\mathbf{R}$ from $\mathbf{O}$ to get the rain-free scene $\mathbf{B}$. This function f above can be represented as a deep neural network and learned by optimizing the loss function $\|f(\mathbf{O}) - \mathbf{R}\|_F^2$.

Based on the motivation above, we proposed the **RE**current **S**E **C**ontext **A**ggregation **N**et (RESCAN) for image deraining. The framework of our network is presented in Fig. 1. We remove rain streaks stage by stage. At each stage, we use the context aggregation network with SE blocks to remove rain streaks. Our network can deal with rain streaks of various directions and shapes, and each feature map in the network corresponds to one kind of rain streak. Dilated convolution used in our network can help us to have large reception field and acquire more contextual information. By using SE blocks, we can assign different weights to various feature maps according to their interdependencies in each convolution layer. As we remove the rain in multiple stages, useful information for rain removal in previous stages can guide the learning in later stages. So we incorporate the RNN architecture with memory unit to make full use of the useful information in previous stages.

In the following, we first describe the baseline model, and then define the recurrent structure, which lifts the model's capacity by iteratively decomposing rain streaks with different characters.

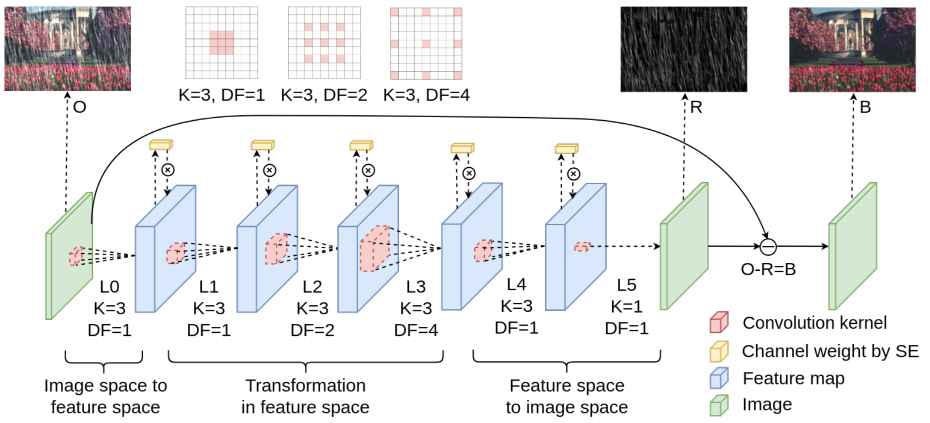

Fig. 2. The architecture of **SE Context Aggregation Network**(SCAN).

Table 1. The detailed architecture of SCAN. d is the depth of network.

Layer	0	1	2	...	d-3	d-2	d-1
Convolution	3×3	3×3	3×3	...	3×3	3×3	1×1
Dilation	1	1	2	...	2^{d-1}	1	1
NonLinear	Yes	Yes	Yes	Yes	Yes	Yes	No
SE block	Yes	Yes	Yes	Yes	Yes	Yes	No
Receptive field	3×3	5×5	9×9	...	$\left(2^{d+1}+1\right)^2$	$\left(2^{d+1}+3\right)^2$	$\left(2^{d+1}+3\right)^2$

4.1 SE Context Aggregation Net

The base model of RESCAN is a forward network without recurrence. We implement it by extending **C**ontext **A**ggregation **N**et(CAN) [11, 12] with Squeeze-and-Excitation (SE) blocks [27], and name it as **SE C**ontext **A**ggregation **N**et (SCAN).

Here we provide an illustration and a further specialization of the SCAN. The SCAN is schematically illustrated in Fig. 2, which is a full-convolution network. In Fig. 2, we set the depth $d = 6$. As large receptive field is very helpful to acquire much contextual information, dilation is adopted in our network. For layers L_1 to L_3, the dilation increases from 1 to 4 exponentially, which leads to the exponential growth in receptive field of every elements. As we treat the first layer as an encoder to transform a image to feature maps, and the last two layers as a decoder to map reversely, we do not apply dilation for layers L_0, L_4 and L_5. Moreover, we use 3×3 convolution for all layers before L_5. To recover RGB channels of a colorful image, or gray channel for a gray image, we adopt the 1×1 convolution for the last layer L_5. Every convolution operation except the last one is followed by a nonlinear operation. The detailed architecture of SCAN is summarized in Table 1. Generally, for a SCAN with depth d, the receptive field of elements in the output image equals to $\left(2^d + 1\right) \times \left(2^d + 1\right)$.

For feature maps, we regard each channel of them as the embedding of a rain streak layer $\mathbf{R}^i$. Recall in Eq. 3, we assign different weights α_i to different rain steak layers $\mathbf{R}^i$. Instead of setting fixed weights α_i for each rain layer, we update the weights for embeddings of rain streak layers in each network layer. Although convolution operation implicitly introduces weights for every channel, these implicit weights are not specialized for every image. To explicitly import weight on each network layer for each image, we extend each basic convolution layer with Squeeze-and-Excitation (SE) block [27], which computes normalized weight for every channel of each item. By multiplying the weights learned by SE block, feature maps computed by convolution are re-weighted explicitly.

An obvious difference between SCAN and former models [8, 25] is that SCAN has no batch normalization (BN) [28] layer. BN is widely used in training deep neural network, as it can reduce internal covariate shift of feature maps. By applying BN, each scalar feature is normalized and has zero mean and unit variance. These features are independent with each other and have the same distribution. However, in Eq. 2, rain streaks in different layers have different distributions in directions, colors and shapes, which is also true for each scalar feature of different rain streak layers. Thus, BN contradicts with the characteristics of our proposed rain model. Therefore, we remove BN from our model. Experimental results in Section 5 show that this simple modification can substantially improve the performance. Furthermore, since BN layers keep a normalized copy of feature maps in GPU, removal of it can sufficiently reduce the usage of GPU memory. For SCAN, we can save approximately 40% of memory usage during training without BN. Consequently, we can build a larger model with larger capacity, or increase the mini-batch size to stabilize the training process.

4.2 Recurrent SE Context Aggregation Net

As there are many different rain streak layers and they overlap with each other, it is not easy to remove all rain streaks in one stage. So we incorporate the recurrent structure to decompose the rain removal into multiple stages. This process can be formulated as:

$$\mathbf{O}_1 = \mathbf{O}, \tag{4}$$

$$\mathbf{R}_t = f\left(\mathbf{O}_t\right), 1 \le t \le T, \tag{5}$$

$$\mathbf{O}_{t+1} = \mathbf{O}_t - \mathbf{R}_t, 1 < t < T, \tag{6}$$

$$\mathbf{R} = \sum_{t=1}^{T} \mathbf{R}_t, \tag{7}$$

where T is the number of stages, $\mathbf{R}_t$ is the output of the t-th stage, and $\mathbf{O}_{t+1}$ is the intermediate rain-free image after the t-th stage.

The above model for deraining has been used in [9, 13]. However, recurrent structure used in their methods [9, 13] can only be regarded as the simple cascade of the same network. They just use the output images of last stage as the input of current stage and do not consider feature connections between these stages.

As these different stages work together to remove the rain, input images of different stages $\{\mathbf{O}_1, \mathbf{O}_2, \cdots, \mathbf{O}_T\}$ can be regarded as a temporal sequence of rain images with decreasing levels of rain streaks interference. It is more meaningful to investigate the recurrent connections between features of different stages rather than only use the recurrent structure. So we incorporate recurrent neural network (RNN) [29] with memory unit to better make use of information in previous stages and guide feature learning in later stages.

For the deraining task, we further explore three different recurrent unit variants, including ConvRNN, ConvGRU [30], and ConvLSTM [31]. In the following subsections, we give a detailed illustration of these three recurrent units evaluated in the experiments.

ConvRNN The first and the simplest unit we examine is RNN [29]. Let x_t and h_t denote the input and hidden states at the t-th stage, respectively. Given the current input x_t and the previous hidden state h_{t-1}, the current cell state h_t can be computed by:

$$h_t = tanh\left(W \circledast x_t + U \circledast h_{t-1} + b\right), \tag{8}$$

where $\circledast$ denotes the convolution operation. W and U are convolutional kernels. h_t is the output of ConvRNN cell. Specifically, W is a dilated kernel introduced in Table 1, and U is a common kernel with size 3×3 or 1×1.

ConvGRU Another unit examined in our framework is ConvGRU. Gated Recurrent Units (GRU) [30] is another commonly used recurrent unit in sequential models. As for its convolutional version ConvGRU, the formulations are:

$$z_t = \sigma\left(W_z \circledast x_t + U_z \circledast h_{t-1} + b_z\right), \tag{9}$$
$$r_t = \sigma\left(W_r \circledast x_t + U_r \circledast h_{t-1} + b_r\right), \tag{10}$$
$$n_t = tanh\left(W_n \circledast x_t + U_n \circledast (r_t \odot h_{t-1}) + b_n\right), \tag{11}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot n_t. \tag{12}$$

where σ is the sigmoid function $\sigma(x) = 1/\left(1 + exp(-x)\right)$. Similar to ConvRNN, Ws and Us are convolutional kernels. The number of parameters for one ConvGRU unit is about 3 times of that for a ConvRNN unit.

ConvLSTM The last recurrent cell we investigate is LSTM [31]. Unlike ConvRNN and ConvGRU, ConvLSTM cell has one more input which is the cell state c_t. Given the current input x_t, the previous cell state c_{t-1} and the previous hiden state h_{t-1}, the current cell state c_t and the current hidden state h_t can be computed as

$$[f, i, o, j]^T = [\sigma, \sigma, \sigma, tanh]^T \left(W \circledast x_t + U \circledast h_{t-1} + b\right), \tag{13}$$
$$c_t = f \odot c_{t-1} + i \odot j, \tag{14}$$
$$h_t = o \odot tanh\left(c_t\right). \tag{15}$$

According to formulations above, we can see that ConvLSTM is more complicated. Compared with ConvRNN, ConvLSTM has 4 times parameters and needs more than 4 times computation for every cell.

4.3 Recurrent framework

We further examine two frameworks to inference the final output. Both of them use $\mathbf{O}_t$ as input for the t-th stage, but they output different images. In the following, we describe the additive prediction and the fully prediction in detail, together with their corresponding loss functions.

Additive prediction The additive prediction is widely used in image processing. In each stage, the network only predicts the residual between previous predictions and the ground truth. It incorporates previous feature maps and states as inputs, which can be formulated as:

$$\mathbf{R}_t = f\left(\mathbf{O}_t, h_{t-1}\right), \tag{16}$$

$$\mathbf{O}_{t+1} = \mathbf{O} - \mathbf{R}_t. \tag{17}$$

where h_{t-1} represents previous states as in Eq. 8, Eq. 12 and Eq. 15. For this framework, the loss functions are listed as follows:

$$L_t\left(\mathbf{\Theta}\right) = \left\|\sum_{j=1}^{t} \mathbf{R}_j - \mathbf{R}\right\|_F^2, 1 \le t \le T \tag{18}$$

$$L\left(\mathbf{\Theta}\right) = \sum_{t=1}^{T} L_t\left(\mathbf{\Theta}\right). \tag{19}$$

where $\mathbf{\Theta}$ represents the network's parameters.

Fully prediction The fully prediction means that in each stage, we predict the whole rain streaks $\mathbf{R}$. This approach can be formulated as:

$$\widehat{\mathbf{R}}_t = f\left(\widehat{\mathbf{O}}_t, h_{t-1}\right), \tag{20}$$

$$\widehat{\mathbf{O}}_{t+1} = \mathbf{O} - \widehat{\mathbf{R}}_t. \tag{21}$$

where $\widehat{\mathbf{R}}_t$ represents the predicted full rain streaks in the t-th stage, and $\widehat{\mathbf{O}}_{t+1}$ equals to $\mathbf{B}$ plus remaining rain streaks. Thus, the corresponding loss functions are:

$$L_t\left(\mathbf{\Theta}\right) = \left\|\widehat{\mathbf{R}}_t - \mathbf{R}\right\|_F^2, 1 \le t \le T \tag{22}$$

$$L\left(\mathbf{\Theta}\right) = \sum_{t=1}^{T} L_t\left(\mathbf{\Theta}\right). \tag{23}$$

5 Experiments

In this section, we present details of experimental settings and quality measures used to evaluate the proposed SCAN and RESCAN models. We compare the performance of our proposed methods with the state-of-the-art methods on both synthetic and real-world datasets.

5.1 Experiment Settings

Synthetic Dataset Since it is hard to obtain a large dataset of clean/rainy image pairs from real-world data, we first use synthesized rainy datasets to train the network. Zhang et al. [25] synthesizes 800 rain images (Rain800) from random selected outdoor images, and splits them into testing set of 100 images and training set of 700 images. Yang et al. [9] collects and synthesizes 3 datasets, including Rain12, Rain100L and Rain100H. We select the most difficult one, Rain100H, to examine our model. It is synthesized with the combination of five streak directions, which causes it hard to effectively remove all rain streaks. There are 1800 synthetic image pairs in Rain100H, and 100 pairs are selected to construct testing set.

Real-world Dataset Both the Rain800 and Rain100H datasets provide many real-world rain images. These images are diverse in terms of content as well as intensity and orientation of rain streaks. We use these datasets for objective evaluation.

Training Settings In training process, we randomly generate 100 patch pairs with size of 64×64 from every training image pairs. The entire network is trained on a Nvidia 1080Ti GPU based on Pytorch framework. We use a batch size of 64 and set the depth of SCAN as $d = 7$ with the receptive field size 35×35. For the nonlinear operation, we use leaky ReLU [32] with $\alpha = 0.2$. For optimization, adam algorithm [33] is adopted with start learning rate 5×10^{-3}. During training, the learning rate is divided by 10 at $15,000$ and $17,500$ iterations.

Quality Measures To evaluate the performance on synthetic image pairs, we adopt two commonly used metrics, including peak signal to noise ratio (PSNR) [34] and structure similarity index (SSIM) [35]. Since there are no ground truth rain-free images for real-world images, the performance on the real-world dataset can only be evaluated visually. We compare our proposed approach with five state-of-the-art methods, including image decomposition (ID) [21], discriminative sparse coding (DSC) [5], layer priors (LP) [7], DetailsNet [8], and joint rain detection and removal (JORDER) [9].

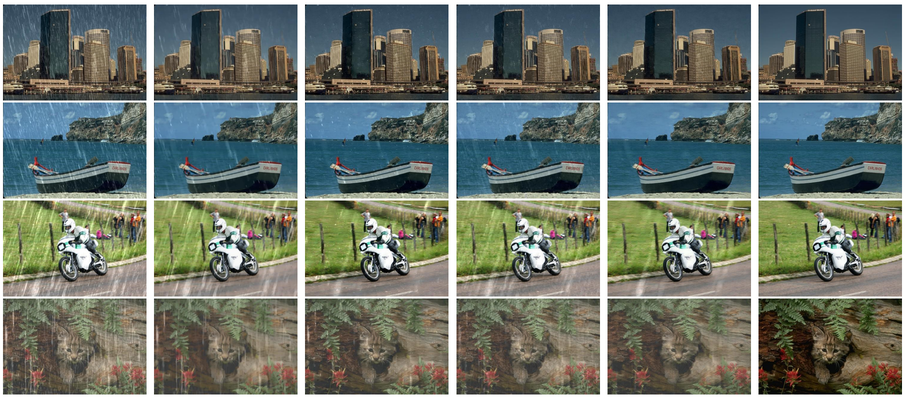

Fig. 3. Results of different methods on synthesized images.

Table 2. Quantitative experiments evaluated on two synthetic datasets.

Dataset	Rain800		Rain100H	
Meassure	PSNR	SSIM	PSNR	SSIM
ID [21]	18.88	0.5832	14.02	0.5239
DSC [5]	18.56	0.5996	15.66	0.4225
LP [7]	20.46	0.7297	14.26	0.5444
DetailsNet [8]	21.16	0.7320	22.26	0.6928
JORDER [9]	22.08	0.7939	22.15	0.6736
JORDER-R [9]	21.82	0.7684	23.45	0.7490
SCAN	23.45	0.8112	22.93	0.7340
RESCAN	**24.09**	**0.8410**	**26.45**	**0.8458**

5.2 Results of Synthetic Data

Table 2 shows results of different methods on the Rain800 and Rain100H datasets. We can see that our RESCAN method considerably outperforms other methods in terms of both PSNR and SSIM on these two datasets. It is also worth noting that our non-recurrent network SCAN can even outperform JORDER and DetailsNet, and is slightly inferior to JORDER-R, the recurrent version of JORDER. This shows the high capacity behind SCAN's shallow structure. Moreover, by using the RNN to gradually recover the full rain streak layer $\mathbf{R}$, RESCAN further improves the performance.

To visually demonstrate the improvements obtained by the proposed methods, results on several difficult sample images are presented in Fig. 3. Please note that we select difficult sample images to show that our method can outperform others especially in difficult conditions, as we design it to deal with complex

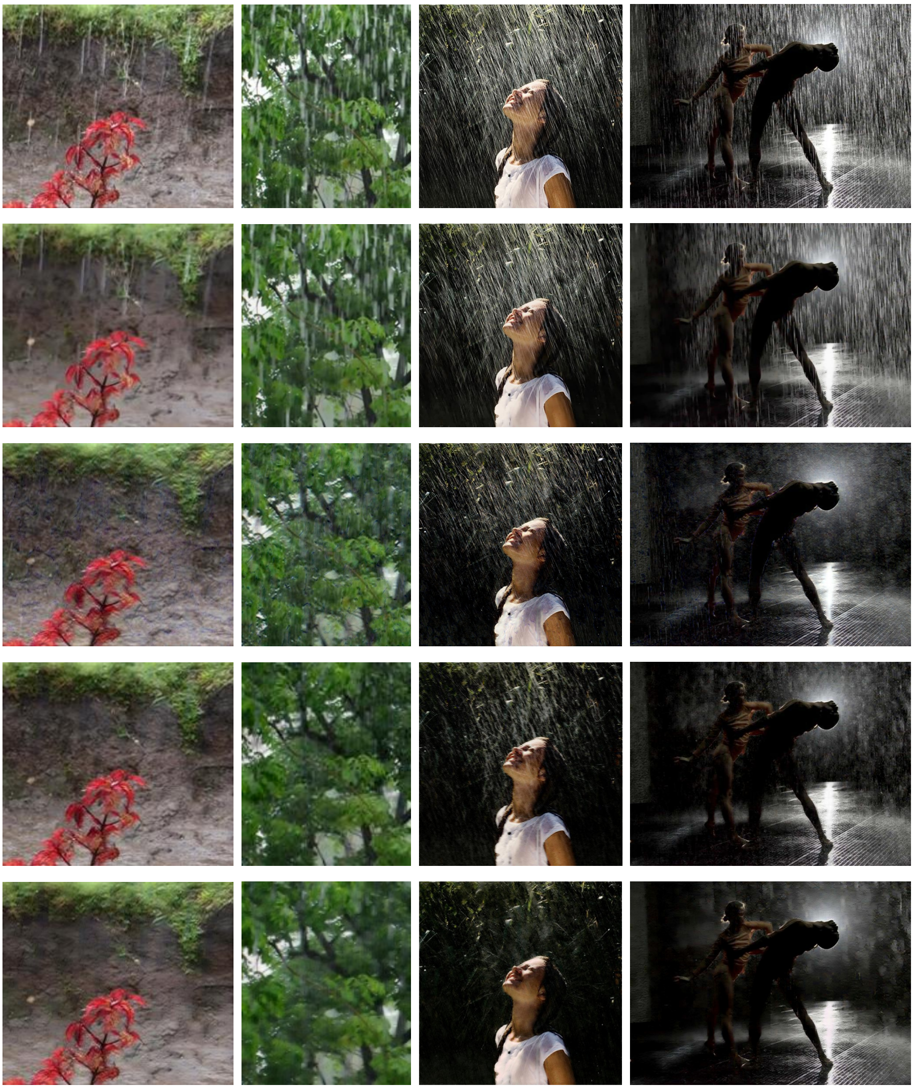

Fig. 4. Results of different methods on real-world images. From top to down: rain image, LP, DetailsNet, JORDER-R and RESCAN.

conditions. According to Fig. 3, these state-of-the-art methods cannot totally remove all rain steaks, while our method can remove the majority of rain steaks as well as maintain details of background scene.

Table 3. Quantitative comparison between SCAN and base model candidates

Dataset	Meassure	Plain	ResNet	EncDec	CAN+BN	CAN	SCAN+BN	SCAN
Rain800	PSNR	22.10	22.10	22.14	22.27	22.45	23.11	**23.45**
	SSIM	0.7816	0.7856	0.7809	0.7871	0.7960	0.7657	**0.8112**
Rain100H	PSNR	21.46	21.51	21.28	22.63	22.93	23.09	**23.56**
	SSIM	0.6921	0.6940	0.6886	0.7256	0.7333	0.7389	**0.7456**

5.3 Results on Real-world Dataset

To test the practicability of deraining methods, we also evaluate the performance on real-world rainy test images. The predictions for all related methods on four real-world sample rain images are shown in Fig. 4. As observed, LP [7] tends to make the output blurry, and DetailsNet [8] tends to add artifacts on derained outputs. We can see that the proposed method can remove most of rain streaks and maintain much texture of background images.

5.4 Analysis of SCAN

To show SCAN is the best choice as a base model for deraining task, we conduct experiments on two datasets to compare the performance of SCAN and its related network architectures, including Plain (dilation=1 for all convolutions), ResNet used in DetailsNet [8] and Encoder-Decoder used in ID-CGAN [25]. For all networks, we set the same depth $d = 7$ and width (24 channels), so we can keep their numbers of parameters and computations the same order of magnitude. More detailedly, we keep layers L_0, L_{d-5} and L_{d-6} of them with same structure, so they only differ in layers L_1 to L_4. The results are shown in Table 3. SCAN and CAN achieve the best performance in all datasets, which can be attributed to the fact that receptive fields of these two methods exponentially increase with the growing of depth, while receptive fields of other methods only have linear relationship with depth.

Moreover, the comparison between results of SCAN and CAN indicates that SE block contributes a lot to the base model, as it can explicitly learn weight for every independent rain streak layer. We also examine the SCAN model with BN. The results clearly verify that removing BN is a good choice in deraining task, as rain steak layers are indenpendent with each others.

5.5 Analysis of RESCAN

As we list three recurrent units and two recurrent frameworks in Section 4.2, we conduct experiments to compare these different settings, consisting of {ConvRNN, ConvLSTM, ConvGRU} $\times$ {Additive Prediction (Add), Full Prediction (Full)}. In Table 4, we report the results. To compare with the recurrent framework in [9, 13], we also implements the settings in Eq. 5 (Iter), in which previous states are not reserved.

Table 4. Quantitative comparison between different settings of RESCAN

Dataset	Rain800		Rain100H	
Meassure	PSNR	SSIM	PSNR	SSIM
Iter+Add	23.36	0.8169	22.81	0.7630
ConvRNN+Add	24.09	0.8410	23.34	0.7765
ConvRNN+Full	23.52	0.8269	23.44	0.7643
ConvGRU+Add	23.31	**0.8444**	24.00	0.7993
ConvGRU+Full	24.18	0.8394	**26.45**	**0.8458**
ConvLSTM+Add	22.93	0.8385	25.13	0.8211
ConvLSTM+Full	**24.37**	0.8384	25.64	0.8334

It's obvious that Iter cannot compete with all RNN structures, as it leaves out information of previous stages. Moreover, ConvGRU and ConvLSTM outperform ConvRNN, as they maintain more parameters and require more computations. However, due to experimental errors and our fixed experiment settings, it is difficult to pick a best unit between ConvLSTM and ConvGRU. For recurrent frameworks, results indicate that Full Prediction is better.

6 Conclusion

We propose the recurrent squeeze-and-excitation based context aggregation network for single image deraining in this paper. We divide the rain removal into multiple stages. In each stage, the context aggregation network is adopted to remove rain streaks. We also modify the CAN to better match the rain removal task, including the exponentially increased dilation and removal of BN layer. To better characteristic intensities of different rain streak layers, we adopt the squeeze-and-excitation block to assign different weights according to their properties. RNN is incorporated to better utilize the useful information for rain removal in previous stages and guide the learning in later stages. We also test the performance of different network architectures and recurrent units. We conduct extensive experiments on both synthetic and real-world datasets. Based on the experimental results, we can see that our proposed method outperforms the state-of-the-art approaches under all evaluation metrics.

References

1. Zhang, X., Li, H., Qi, Y., Leow, W.K., Ng, T.K.: Rain removal in video by combining temporal and chromatic properties. In: IEEE ICME. (2006) 461–464
2. Garg, K., Nayar, S.K.: Vision and rain. International Journal of Computer Vision **75**(1) (2007) 3–27
3. Santhaseelan, V., Asari, V.K.: Utilizing local phase information to remove rain from video. International Journal of Computer Vision **112**(1) (2015) 71–89
4. Tripathi, A.K., Mukhopadhyay, S.: Removal of rain from videos: a review. Signal, Image and Video Processing **8**(8) (2014) 1421–1430
5. Luo, Y., Xu, Y., Ji, H.: Removing rain from a single image via discriminative sparse coding. In: IEEE ICCV. (2015) 3397–3405
6. Chang, Y., Yan, L., Zhong, S.: Transformed low-rank model for line pattern noise removal. In: IEEE ICCV. (2017) 1726–1734
7. Li, Y., Tan, R.T., Guo, X., Lu, J., Brown, M.S.: Rain streak removal using layer priors. In: IEEE CVPR. (2016) 2736–2744
8. Fu, X., Huang, J., Zeng, D., Huang, Y., Ding, X., Paisley, J.: Removing rain from single images via a deep detail network. In: IEEE CVPR. (2017) 1715–1723
9. Yang, W., Tan, R.T., Feng, J., Liu, J., Guo, Z., Yan, S.: Deep joint rain detection and removal from a single image. In: IEEE CVPR. (2017) 1357–1366
10. Huang, D.A., Kang, L.W., Yang, M.C., Lin, C.W., Wang, Y.C.F.: Context-aware single image rain removal. In: IEEE ICME. (2012) 164–169
11. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
12. Chen, Q., Xu, J., Koltun, V.: Fast image processing with fully-convolutional networks. In: IEEE ICCV. (2017) 2516–2525
13. Li, R., Cheong, L.F., Tan, R.T.: Single image deraining using scale-aware multi-stage recurrent network. arXiv preprint arXiv:1712.06830 (2017)
14. Barnum, P.C., Narasimhan, S., Kanade, T.: Analysis of rain and snow in frequency space. International Journal of Computer Vision **86**(2-3) (2010) 256
15. Garg, K., Nayar, S.K.: Detection and removal of rain from videos. In: IEEE CVPR. Volume 1. (2004) 528–535
16. Garg, K., Nayar, S.K.: When does a camera see rain? In: IEEE ICCV. Volume 2. (2005) 1067–1074
17. Bossu, J., Hautière, N., Tarel, J.P.: Rain or snow detection in image sequences through use of a histogram of orientation of streaks. International Journal of Computer Vision **93**(3) (2011) 348–367
18. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online dictionary learning for sparse coding. In: ICML. (2009) 689–696
19. Reynolds, D.A., Quatieri, T.F., Dunn, R.B.: Speaker verification using adapted gaussian mixture models. Digital Signal Processing **10**(1-3) (2000) 19–41
20. Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., Ma, Y.: Robust recovery of subspace structures by low-rank representation. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(1) (2013) 171–184
21. Kang, L.W., Lin, C.W., Fu, Y.H.: Automatic single-image-based rain streaks removal via image decomposition. IEEE Transactions on Image Processing **21**(4) (2012) 1742–1755
22. Wang, Y., Liu, S., Chen, C., Zeng, B.: A hierarchical approach for rain or snow removing in a single color image. IEEE Transactions on Image Processing **26**(8) (2017) 3936–3950

23. Gu, S., Meng, D., Zuo, W., Zhang, L.: Joint convolutional analysis and synthesis sparse representation for single image layer separation. In: IEEE ICCV. (2017) 1717–1725
24. Zhu, L., Fu, C.W., Lischinski, D., Heng, P.A.: Joint bi-layer optimization for single-image rain streak removal. In: IEEE CVPR. (2017) 2526–2534
25. Zhang, H., Sindagi, V., Patel, V.M.: Image de-raining using a conditional generative adversarial network. arXiv preprint arXiv:1701.05957 (2017)
26. Kaushal, H., Jain, V., Kar, S.: Free-space optical channel models. In: Free Space Optical Communication. Springer (2017) 41–89
27. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507 (2017)
28. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. (2015) 448–456
29. Mandic, D.P., Chambers, J.A., et al.: Recurrent neural networks for prediction: learning algorithms, architectures and stability. Wiley Online Library (2001)
30. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
31. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)
32. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: ICML. Volume 30. (2013) 3
33. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
34. Huynh-Thu, Q., Ghanbari, M.: Scope of validity of psnr in image/video quality assessment. IET Electronics letters **44**(13) (2008) 800–801
35. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4) (2004) 600–612