Supplementary Material for Recurrent Squeeze-and-Excitation Context Aggregation Net for Single Image Deraining

Xia Li[123⋆], Jianlong Wu[23⋆], Zhouchen Lin[23], Hong Liu[1(✉)], Hongbin Zha[23]

[1]Key Laboratory of Machine Perception, Shenzhen Graduate School, Peking University
[2]Key Laboratory of Machine Perception (MOE), School of EECS, Peking University
[3]Cooperative Medianet Innovation Center, Shanghai Jiao Tong University
{ethanlee,jlwu1992,zlin,hongliu}@pku.edu.cn, zha@cis.pku.edu.cn

Abstract. In this document, we provide more experimental analysis and results on synthesized and real-world images. All figures are best viewed at screen!

1 Various Recurrent Units

For the deraining task, we explore three different recurrent unit variants, including ConvRNN, ConvGRU [1], and ConvLSTM [2]. In the following, we give a detailed illustration of these three recurrent units evaluated in the experiments. Here, we denote x_s^j as the feature map of the j-th layer in the s-th stage, which can be computed based on the x_{s-1}^j (feature maps in the same layer of the previous stage) and x_s^{j-1} (feature maps in the previous layer of the same stage). W^j and U^j are convolutional kernels of the j-th layer for all stages. Specifically, in our RESCAN model, W is a dilated kernel, and U is a common kernel with a size 3×3 or 1×1.

ConvRNN RNN [3] is the simplest convolutional unit. Its convolutional version can be formulated as:

$$x_s^j = \tanh\left(W^j \circledast x_s^{j-1} + U^j \circledast x_{s-1}^j + b^j\right), \tag{1}$$

where $\circledast$ denotes the convolution operation.

ConvGRU Gated Recurrent Units (GRU) [1] is one of the most commonly used recurrent units in sequential models. Its convolutional version ConvGRU is

⋆ Equal contributions

adopted in our model:

$$z_s^j = \sigma\left(W_z^j \circledast x_s^{j-1} + U_z^j \circledast x_{s-1}^j + b_z^j\right), \tag{2}$$

$$r_s^j = \sigma\left(W_r^j \circledast x_s^{j-1} + U_r^j \circledast x_{s-1}^j + b_r^j\right), \tag{3}$$

$$n_s^j = \tanh\left(W_n^j \circledast x_s^{j-1} + U_n^j \circledast \left(r_s^j \odot x_{s-1}^j\right) + b_n^j\right), \tag{4}$$

$$x_s^j = \left(1 - z_s^j\right) \odot x_{s-1}^j + z_s^j \odot n_s^j, \tag{5}$$

where σ is the sigmoid function $\sigma(x) = 1/(1+\exp(-x))$ and $\odot$ denotes element-wise multiplication.

ConvLSTM The last recurrent cell we investigate is LSTM [2]. Unlike ConvRNN and ConvGRU, ConvLSTM cell has one more input which is the cell state c_s^j. For the j-th layer of the s-th stage, given the current input x_s^{j-1}, the previous cell state c_{s-1}^j and the previous hiden state x_{s-1}^j, the current cell state c_s^j and the current hidden state x_s^j can be computed as:

$$f_s^j = \sigma\left(W_f^j \circledast x_s^{j-1} + U_f^j \circledast x_{s-1}^j + b_f^j\right), \tag{6}$$

$$i_s^j = \sigma\left(W_i^j \circledast x_s^{j-1} + U_i^j \circledast x_{s-1}^j + b_i^j\right), \tag{7}$$

$$o_s^j = \sigma\left(W_o^j \circledast x_s^{j-1} + U_o^j \circledast x_{s-1}^j + b_o^j\right), \tag{8}$$

$$t_s^j = \tanh\left(W_t^j \circledast x_s^{j-1} + U_t^j \circledast x_{s-1}^j + b_t^j\right), \tag{9}$$

$$c_s^j = f_s^j \odot c_{s-1}^j + i_s^j \odot t_s^j, \tag{10}$$

$$x_s^j = o_s^j \odot \tanh\left(c_s^j\right). \tag{11}$$

Compared with a convolutional unit, ConvRNN, ConvGRU and ConvLSTM units will have twice, three times and four times parameters, respectively.

2 Influence of Parameters

In this section, we conduct experiments to compare the performance of different methods with similar number of parameters.

All stages in RESCAN share the same set of parameters, so the number of parameters is unrelated to the number of stages. However, with RNN units, RESCAN surely has more parameters than SCAN. But they are still comparable. As we stated in Section 4.2 of the paper, assume all convolutions have the same kernel size, such as 3×3, then compared with a Conv unit, ConvRNN, ConvGRU and ConvLSTM units will have twice, three times and four times parameters, respectively. But we find that not every convolution kernel in RNN unit plays an important role. Only the W in Eq. (1), the W_n in Eq. (4) and the W_j in Eq. (9) are the most important ones in their units. So we can largely reduce the

Table 1: Quantitative experiments evaluated on two synthetic datasets. '+' and '−' denote the changes in number of channels. '1×1' represents setting the size of unimportant convolution kernels to 1×1.

Dataset	Rain800		Rain100H		Param
Meassure	PSNR	SSIM	PSNR	SSIM	Param
DetailsNet− [8]	21.75	0.7422	22.04	0.6828	17.7M
JORDER [9]	22.24	0.7763	22.15	0.6736	17.8M
JORDER-R [9]	22.29	0.7922	23.45	0.7490	71.3M
SCAN+	23.03	0.8110	24.28	0.7731	17.6M
RESCAN (1×1)	24.13	0.8300	25.52	0.8215	17.1M
RESCAN (3×3)	**24.09**	**0.8410**	**26.45**	**0.8458**	59.9M

Table 2: Voting results of DetailsNet, JORDER-R and RESCAN on real images. 'Selected' represents the number of most voted selection.

Guideline	Best Derain		Best Detail	
Meassure	Voted	Selected	Voted	Selected
Not Sure	189	49	231	56
DetailsNet [4]	40	1	50	0
JORDER-R [5]	150	21	150	14
RESCAN	**383**	**65**	**331**	**57**

number of parameters by reducing other kernels' sizes to 1×1. The rate of their parameters number among different methods becomes SCAN:RESCAN (ConvRNN):RESCAN (ConvGRU):RESCAN (ConvLSTM)=9:10:14:16. To further make the parameters number of SCAN consistent with that of RESCAN (ConvGRU), we increase the channels number of SCAN. We also decrease the channels number of DetailsNet [8] for fair comparison.

Experimental results are presented in Table 1. We can see that with roughly the same number of parameters, RESCAN (1×1) outperforms DetailsNet, JORDER and SCAN, which can prove that RESCAN's improvement over SCAN does not arise from the more parameters, but the usage of RNN structure.

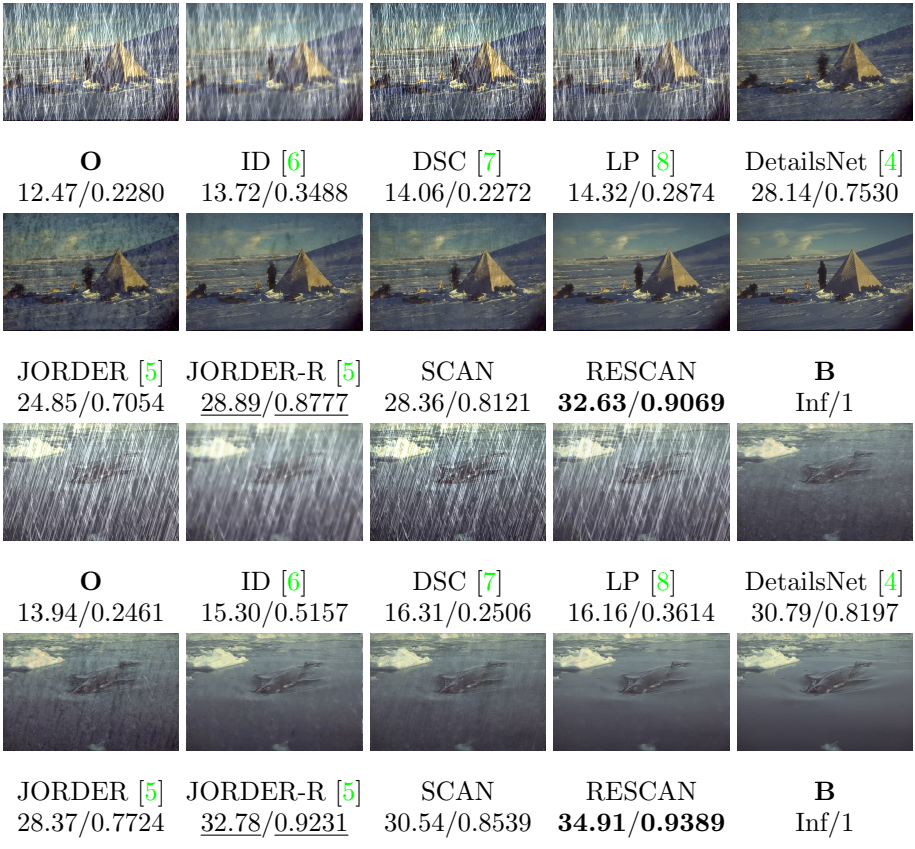
3 User Study on Real-world Images

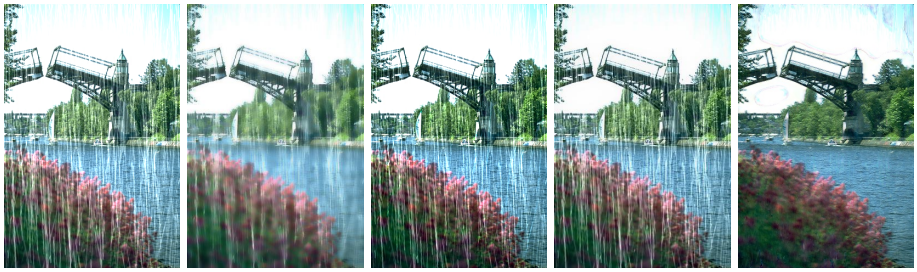
To further validate the effectiveness of our method on real-world dataset, we make a user study based on the results of real-world deraining images.

Rain800 and Rain100H contain 67 real rainy images in total. We also select 50 most related images from top 100 results by searching 'rain' in *Google Images*. Based on the output results of DetailsNet [4], JORDER-R [5] and RESCAN on these 127 real images, we invite 6 people to select the one with the least rain streaks, and the one preserving most texture details, or not sure. Results are

shown in Table 2. We can see that the votes of RESCAN under both deraining and detail preserving criteria are twice more than those of JORDER-R [5], which is also true for the number of selected images. It can well demonstrate our superiority over other methods on real images.

4 More Results on Synthetic Data



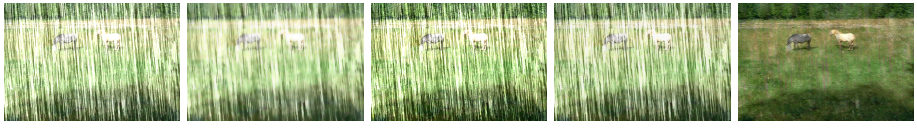


O	ID [6]	DSC [7]	LP [8]	DetailsNet [4]
16.00/0.6057	17.00/0.6062	17.91/0.6203	17.44/0.6099	25.90/0.8642

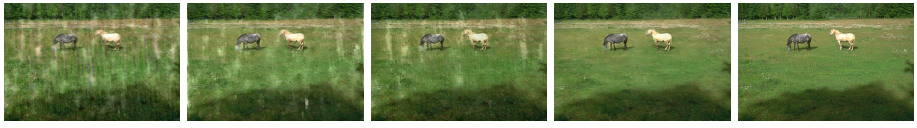
JORDER [5]	JORDER-R [5]	SCAN	RESCAN	**B**
23.97/0.8353	<u>27.94</u>/<u>0.9233</u>	26.51/0.8734	**29.38/0.9251**	Inf/1

O	ID [6]	DSC [7]	LP [8]	DetailsNet [4]
12.29/0.2546	13.37/0.4249	13.40/0.2444	13.91/0.3054	26.63/0.6435

JORDER [5]	JORDER-R [5]	SCAN	RESCAN	**B**
23.80/0.6133	<u>30.45</u>/<u>0.8738</u>	27.82/0.6885	**33.94/0.8790**	Inf/1



O	ID [6]	DSC [7]	LP [8]	DetailsNet [4]
8.75/0.2015	9.23/0.2305	9.71/0.2062	9.86/0.1900	23.89/0.6689



JORDER [5]	JORDER-R [5]	SCAN	RESCAN	**B**
22.47/0.6322	23.22/<u>0.7193</u>	<u>24.20</u>/0.6985	**27.27/0.7580**	Inf/1

O	ID [6]	DSC [7]	LP [8]	DetailsNet [4]
6.99/0.2369	7.14/0.2586	7.27/0.2325	7.59/0.2127	17.88/0.5335

JORDER [5]	JORDER-R [5]	SCAN	RESCAN	**B**
15.190/0.4838	18.13/0.6122	17.58/0.5095	**21.06/0.6962**	Inf/1

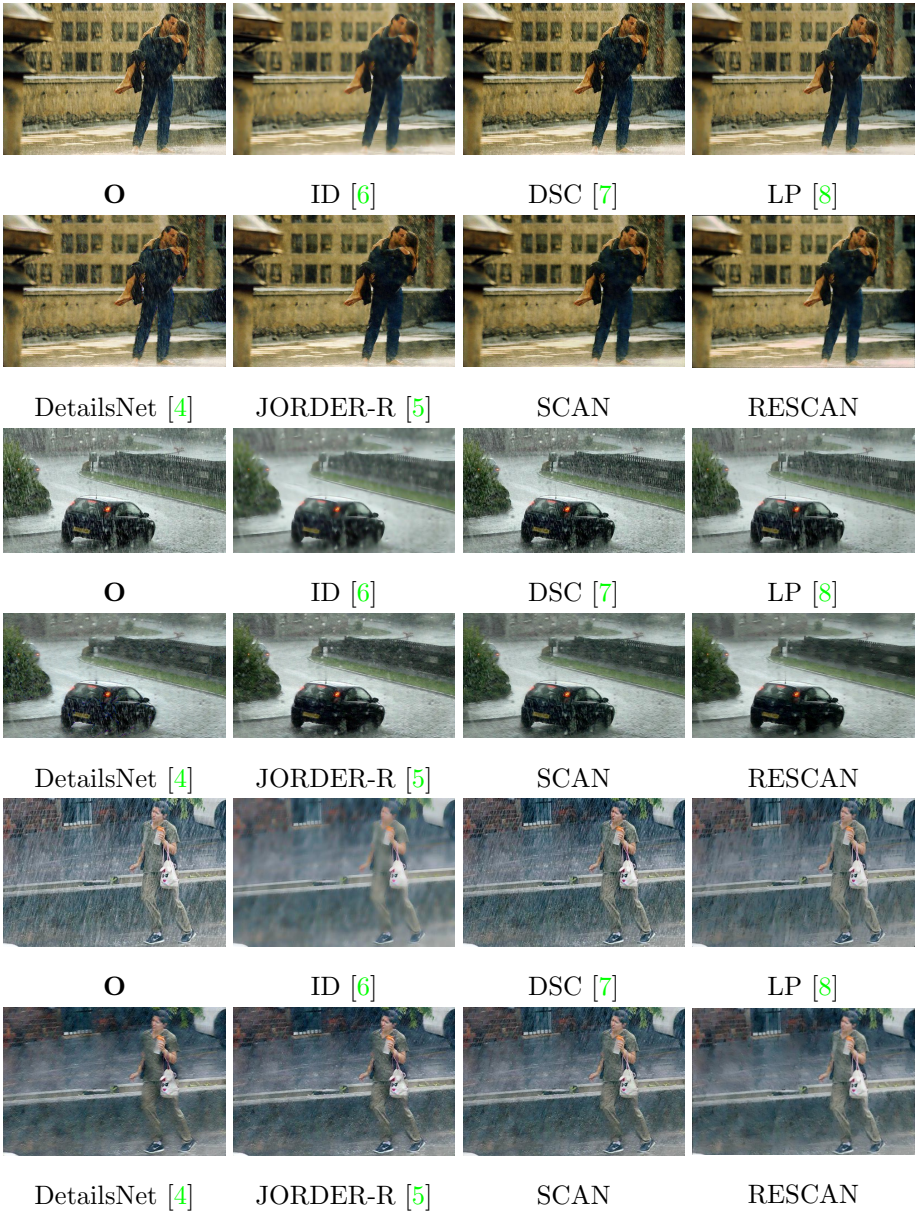
O	ID [6]	DSC [7]	LP [8]	DetailsNet [4]
10.19/0.3024	10.75/0.3256	12.19/0.3372	11.30/0.3110	22.10/0.7125

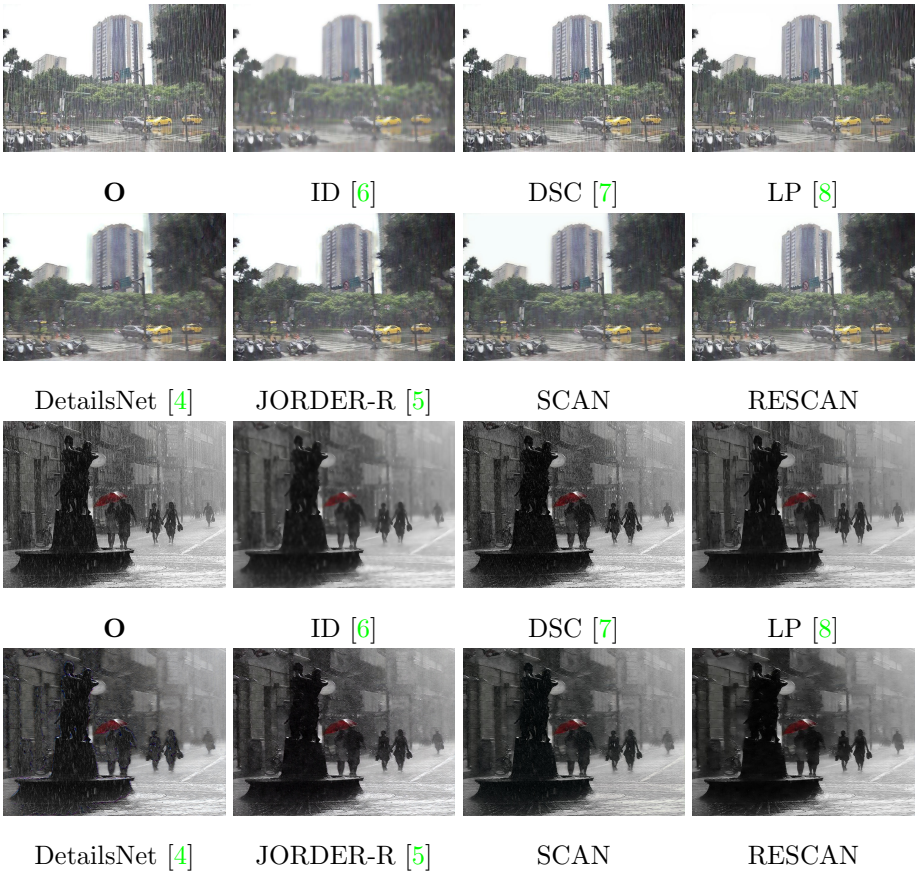
JORDER [5]	JORDER-R [5]	SCAN	RESCAN	**B**
21.57/0.6733	23.09/0.7797	23.43/0.7342	**26.55/0.8565**	Inf/1

O	ID [6]	DSC [7]	LP [8]	DetailsNet [4]
9.83/0.2831	10.42/0.3474	11.60/0.3044	10.97/0.2690	22.22/0.6744

JORDER [5]	JORDER-R [5]	SCAN	RESCAN	**B**
21.51/0.6375	23.12/0.7862	23.90/0.6676	**27.24/0.8372**	Inf/1

5 More Results on Real-world Data


O
ID [6]
DSC [7]
LP [8]
DetailsNet [4]
JORDER-R [5]
SCAN
RESCAN
O
ID [6]
DSC [7]
LP [8]
DetailsNet [4]
JORDER-R [5]
SCAN
RESCAN
O
ID [6]
DSC [7]
LP [8]
DetailsNet [4]
JORDER-R [5]
SCAN
RESCAN



References

1. 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)
2. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)
3. Mandic, D.P., Chambers, J.A., et al.: Recurrent neural networks for prediction: learning algorithms, architectures and stability. Wiley Online Library (2001)
4. 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
5. 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
6. 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

7. Luo, Y., Xu, Y., Ji, H.: Removing rain from a single image via discriminative sparse coding. In: IEEE ICCV. (2015) 3397–3405
8. Li, Y., Tan, R.T., Guo, X., Lu, J., Brown, M.S.: Rain streak removal using layer priors. In: IEEE CVPR. (2016) 2736–2744