Deep learning based Deraining Approach Comparison and Analysis

Yutong Xie[1], Weiran Guan[1], Yunyi Zhang[1], Junhong Chen[1]

[1]University of Illinois at Urbana Champaign

Abstract

Deraining is a hot topic in computer vision area, and it is very useful in our daily life. In recent years, many people proposed different approaches to realize the deraining and achieved great progress. Existing methods can be divided into two categories, including video-based methods and single image removal methods. For single image removal, traditional methods, such as discriminative sparse coding, low rank representation, and Gaussian mixture model, can be applied to this task, and have a good performance. Recently, the deep learning-based elimination method has received widespread attention due to its powerful feature representation ability. But all these related methods still have a lot of room for improvement. In this project, we will compare different single image deraining approach and conduct corresponding experiments. The project narrated video was uploaded to Youtube with the URL: https://youtu.be/AaXoVwj_yKM.

1. Introduction

Rain introduces visual degradations to captured images and videos. Rain streaks particularly in heavy rain can cause severe occlusion on the background scene. Human vision and many computer vision algorithms suffer from this degradation, since most of these algorithms assume clear weather, with no interference of rain streaks and rain accumulation. In the past few decades, lots of algorithms were proposed to solve the problem, which include both traditional optimization based methods and deep learning based approaches.

Before 2017 , the typical methods for deraining are model-based approach. Model-based methods rely more on the statistical analysis of rain streaks and background scenes. However since 2017, single-image deraining methods enter into a period of data driven approach. Many different data driven approaches are proposed for image derain such as,deep convolutional network, generative adversarial network, and semi/unsupervised methods. These methods exploit deep networks to automatically extract hierarchical features, enabling them to model more complicated mappings from rain images to clean ones. Architecture wise, some methods utilize recurrent network, or recursive network to remove rain progressively.

In this project, we chose two methods and compared their effects on different datasets. The approach performance will be related to PSNR and SSIM. During the literature review process, we found that there are several novel approaches provided by scholars these years, such as PreNet, RESCAN, SPANet, and so on. We read those papers that describe those algorithms and familiarize ourselves with the preprocesses and the network structures in the papers. After reading several papers, we decided to choose PreNet and RESCAN and compare the performance of those two. Because the architectures of Progressive ResNet (PRN) and PReNet are similar, we compared PRN and PreNet as well. During our experiments, we trained and tested the same sets of data on both algorithms and collected and analysed the result for all of them.

2. Network Architecture

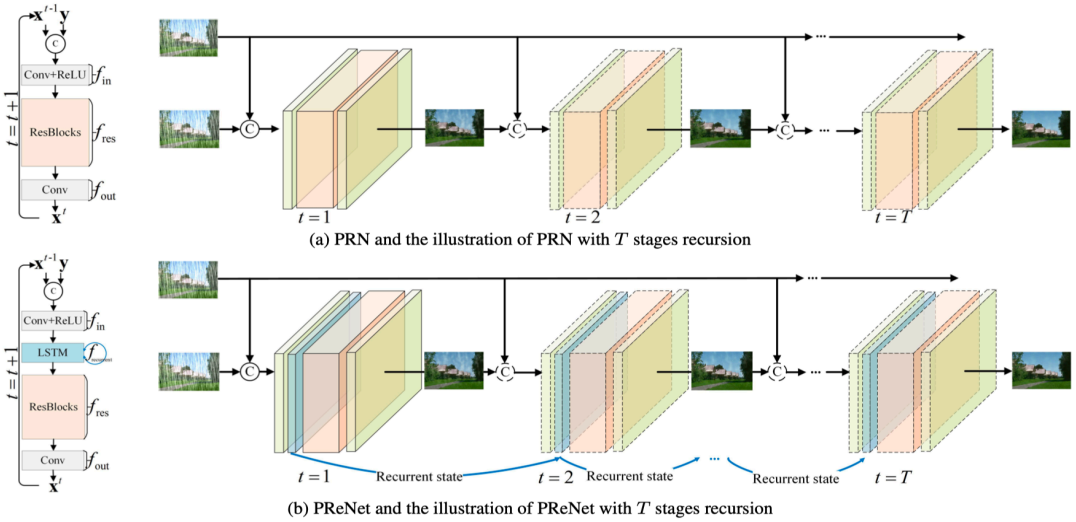
In the part, we will introduce network architecture of progressive network and RESCAN. Progressive network includes Progressive ResNet (PRN), PReNet, and their extension.

2.1 Progressive Network

Progressive Image Deraining Network(PreNet) is proposed by DongWei from China in 2019. Progressive ResNet(PRN) is proposed to take advantage of recursive computation by repeatedly unfolding a shallow Resnet. PreNet is created based on the recurrent layer. By utilizing recursive ResNet, network parameters can be reduced without degrading the performance.

PreNet begins with a basic shallow ResNet with five residual blocks. Then, PRN involved by unfolding the ResNet recursively into multiple stages. PreNet adds one more convolutional LSTM layer (a recurrent layer) to generate the dependencies of deep features across different

stages based on PRN. Then, the detailed architectures of PRN and PreNet will be introduced.



(a) PRN and the illustration of PRN with T stages recursion

(b) PReNet and the illustration of PReNet with T stages recursion

Figure 1. The architectures of progressive networks. f_{in} is a convolution layer with ReLU, f_{res} is ResBlocks, f_{out} is a convolution layer, $f_{recurrent}$ is a convolutional LSTM.

2.1.1 Progressive Residual Network (PRN)

Progressive residual network (PRN) includes three different parts as shown in Fig. 1(a): (i) a convolutional layer f_{in} to receive inputs, (ii) several residual blocks f_{res} to extract input features, (iii) a convolutional layer f_{out} provides the output. We can easily see the inference of PRN at stage t here.

$$x' = f_{in}(x^{t-1}, y)$$
$$x^t = f_{out}(f_{res}(x'))$$

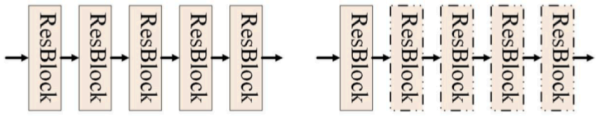
One thing we need to point out here is that fin takes both x_{t-1} and y as the inputs, where y means the original rainy image. The author tested the performance of both including y or not and found that the deraining performance was better with the addition of raw image.

2.1.2 Progressive Recurrent Network

PreNet is developed based on PRN by adding a recurrent layer to introduce feature dependencies across stages, which is helpful for removing rain streak. The recurrent layer can be implemented by Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU). In PreNet, the former one was taken. Then, the inference of PreNet is shown below. The recurrent layer takes both s_{t-1} and x' as input.

$$x' = f_{in}(x^{t-1}, y)$$
$$s^t = f_{recurrent}(s^{t-1}, x')$$
$$x^t = f_{out}(f_{res}(s^t))$$

For both PRN and PreNet, the filters are with kernel size 3x3 and padding size 1x1. In the network, f_{in} is a 1-layer convolution followed by a ReLU layer, f_{res} includes 5 ResBlocks, and f_{out} is a 1-layer convolution. Because PRN and PreNet take both x_{t-1} and y as the input, so fin should be with 6 channels input totally and 32 channels output. For f_{res}, it will take 32 input channels and 32 output channels, and f_{out} will take 32 input channels and output 3 channels. There are two different implementations for f_{res}, conventional ResBlocks and recursive Resblocks.



(a) Conventional ResBlocks (b) Recursive ResBlocks

Figure 2. Implementations of f_{res}: (a) conventional ResBlocks and (b) recursive Resblocks

Conventional ResBlocks: For conventional ResBlocks, it just simply concatenated by 5 continuous ResBlocks. Each ResBlocks constituted 2 convolutional layers and one ReLU layer. All the layers are with 32 input channels and output channels. PRN and PreNet use conventional ResBlocks.

Recursive ResBlocks: For recursive ResBlock, it unfolds one ResBlock five times as shown in Fig. 2(b). Compared to conventional ResBlocks, recursive methods decrease the number of parameters which save great computational resources. By substituting conventional ResBlocks by recursive ResBlocks, we can get PRN_r and $PreNet_r$. In the

experiment section, we test the performance of these two networks, so we will not talk more here.

2.2 RESCAN

In RESCAN, the author also indicated the rain model for us to understand how to conduct deraining work better. Then, we will introduce the rain model first and network architectures later.

2.2.1 Rain Models

R is composed of rain streaks. In addition, the modeling goals of the model are also different. This article hopes to model to get R, so that B = O - R. Therefore, the overall goal of this article is to learn a mapping F so that F (O) and R are as close as possible.

To reduce the complexity, 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 an unpolluted background. Based on this, the rain model can be reformulated as follows:

$$O = B + \sum_{i=1}^{n} R^i$$

where R^i represents the i-th rain streak layer that consists of one kind of rain streaks and n is the total number of different rain streak layers.

Accumulation of multiple rain streaks in the air may cause attenuation and scattering, which further increases the diversity of rain streaks' brightness. This further pollutes the observed image O. For camera imaging, due to the limitation of pixel number, far away rain streaks cannot occupy full pixels. To handle the issues above, the researchers further take the global atmospheric light into consideration and assign different alpha-values to various rain streak layers according to their intensities transparencies. Further generalize the rain model to:

$$O = \left(1 - \sum_{i=0}^{n} \alpha_i\right) B + \alpha_0 A + \sum_{i=1}^{n} \alpha_i R^i, s.t. \alpha_i \geq 0, \sum_{i=0}^{n} \alpha_i \leq 1$$

where A is the global atmospheric light, α_0 is the scene transmission, α_i (i = 1, · · · , n) indicates the brightness of a rain streak layer or a haze layer.

2.2.2 Network Design

The article proposes a framework called *Recurrent SE Context Aggregation Net* (RESCAN). The whole process is multi-stage, and R.

Each stage uses a structure called SCAN as shown below. And the completed network architecture is shown in Fig. 4.

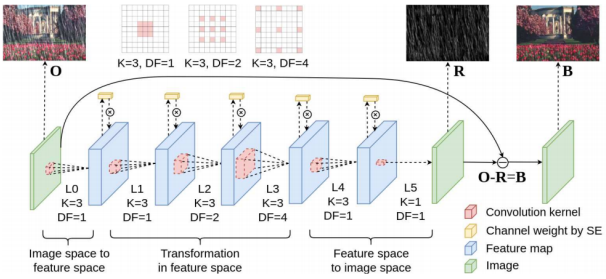


Figure 3. The architecture of SE Context Aggregation Network (SCAN).

This structure has the following characteristics:

- The first is to use dilated convolution to quickly expand the perception field, because more context information is more critical in the rain, such as the multi-scale Loss of the previous article.

- The second feature is the use of the SE module. The author believes that each channel of the feature map can be regarded as a representation (embedding) of some R_i. Each R_i has a corresponding coefficient α_i. By introducing the SE module, you can explicitly Assign different coefficients to different R_i. In this way, certain rain streaks can be obtained in each stage, which means that some rain streaks can be removed in each stage.

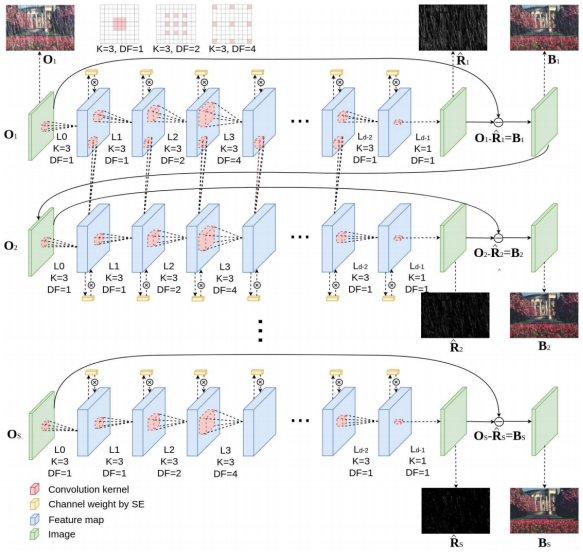


Figure. 4. The unfolded architecture of RESCAN

3. Comparison of Algorithms

In the RESCAN algorithm, the network has a much more complex network structure. In this network, the model derains by using a lot of stages to eliminate the rain streak step by step and using many channels to handle different rain streak shapes. In each stage, the uses context aggregation network from the SE Context Aggregation Network (SCAN) which shows in the Figure below and multiple fully- connected networks. The total network is shown in Fig. In the base model, SCAN is a forward network without recurrence, and it is a fully connected network. Compared with the normal network model, there is no batch normalized layer in the SCAN network, which makes this model less memory consumable. In RESCAN, per the reference, it uses the convolutional Gated Recurrent Units (ConvGRU) as the recurrent units.

In PRN, the model does not have the Squeeze-and-Excitation units like RESCAN model and the SCAN model. It uses the basic parts in the ResNet: (1) convolutional input layer to take input and output at the start and the end of the pipeline. (2) residual blocks (ResBlocks) to get the features. In this model, ResBlocks includes 2 convolution layers followed by ReLU. And this network repeats this for 5 times. In this model the structure is shown in Fig. In PReNet, there is a recurrent unit to help increase performance of the network. Compared to the RESCAN model, this model uses the convolutional Long Short-Term Memory (LSTM), rather than GRU method. The model structure is shown in Fig.

Compared with using a deeper network to complete the derain mission, the PRN and PReNet use a recursive way to eliminate the rain streak in the pictures. In this way, the network seems to be simpler and more efficient.

4. Experiment

4.1 Sythetic 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.

- **Rain800.** 800 rain images from randomly selected outdoor images, and split them into a testing set of 100 images and training set of 700 images.
- **Rain12.** 12 images of outdoors generated with random rain streaks on them. This dataset is not used for training but only for testing trained models.
- **Rain100L and Rain100H**. Rain100H is synthesized with the combination of five streak directions, which makes it hard to effectively remove all rain streaks. There are 1, 800 synthetic image pairs in RainTrainH, and 100 pairs(Rain100H) are selected as the testing set.
- **Rain12600 and Rain1400.** Similar to the images in Rain100H dataset, this dataset has more images which up to 12600 images in the training set and 1400 images for us to test our models.
- **Attentive-GAN.** This dataset is from a paper named *Attentive Generative Adversarial Network for Raindrop Removal from A Single Image.* Unlike the synthetic dataset above, this dataset is real world images. The main difference is that this dataset is about the raindrop on the glass or on the camera lens. The situation is totally different from the generated rain streaks.

4.2 Progressive Network Training Settings

The networks are implemented by using Pytorch. We processed our training and testing part on Google Colab with one NVIDIA P100 GPU. In the experiments, I set the patch size as 100x100, and the batch size as 18. For the training process, we use the Adam algorithm and the initial learning rate is 1x10-3. The total training epoch is set as 100 for all models. Also, we set the scheduler to adjust the learning rate with gamma 0.2 when epochs achieve 30, 50, and 80. And the progressive networks are trained with SSIM loss. We conducted several different training and testing experiments on different datasets.

Training Dataset	Test Dataset
TrainRainL	Rain100L, Rain12
TrainRainH	Rain100H, Rain12
TrainRain12600	Rain1400, Rain12
Rain800(Train)	Rain800(Test)
AttentiveGAN-Data(Train)	AttentiveGAN-Data(Test)

Table1. The matches of training dataset and test dataset.

4.3 RESCAN Training Settings

In the training process, we randomly generate 100 patch pairs with a size of 64 × 64 from every training image pair. The entire network is trained on an Nvidia P100 GPU based on Pytorch. 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 with $\alpha = 0.2$. For optimization, the ADAM algorithm is adopted with a start learning rate 5×10^{-3} .

4.4 Quality Measures

To evaluate the performance on synthetic image pairs, we adopt two commonly used metrics, including peak signal to noise ratio (PSNR) and structure similarity index

(a) Rainy image (b) PRN (c) PRN_r

(d) PreNet (e) PReNet_r (f) RESCAN

Figure 5. Visual effects of different deraining algorithm on Rain100L dataset.

Dataset	Metric	PRN	PRN_r	PreNet	PreNet_r	RESCAN
Rain100L	PSNR	36.9854	36.1051	**37.4800**	37.1043	36.0590
	SSIM	0.9772	0.9725	**0.9792**	0.9773	0.9355
Rain100H	PSNR	28.0703	27.4266	29.4647	28.9788	**29.5116**
	SSIM	0.8838	0.8735	**0.8990**	0.8921	0.8845
Rain800	PSNR	25.7509	25.5858	26.2700	26.0198	**32.6824**
	SSIM	0.8822	0.8846	**0.8906**	0.8902	0.8055
Rain1400	PSNR	31.6915	31.3139	**32.5950**	32.4445	32.0664
	SSIM	0.9408	0.9365	**0.9459**	0.9444	0.8686
Rain12	PSNR	32.3536	32.5591	32.2095	**32.6241**	32.3878
	SSIM	0.9205	0.9323	0.9104	**0.9195**	0.8765
AttentiveGAN-Data	PSNR	24.3479	21.4914	24.4930	24.9112	**29.4526**
	SSIM	0.8752	0.8373	0.8725	**0.8805**	0.7596

Table 2. Average PSNR and SSIM comparison on the synthetic rain streak dataset Rain100L, Rain100H, Rain800, Rain1400, Rain12, and raindrop dataset AttentiveGAN-Data.

(SSIM). We measured PSNR for each predicted image with its ground truth image and then calculated the average PSNR over the whole test image set. We also calculate the average SSIM over each dataset's test set.

5. Results and Analysis

After training and testing on different datasets with different algorithms, we get the results as shown in the table 2. And we attched some test results in Fig. 5 for better visualization.

5.1 RainStreak vs RainDrop

From the table 2, we can see that all networks perform poorly on AttentiveGAN-Data.

And Fig. 6 shows that the deraing image looks not good. The possible reason is that the AttentiveGAN dataset includes all rainy images with raindrop, while others includes images with rain streak. Rain streak model can be seen as the linear addition of clean background B and sparse linear rain R as we mentioned before. However, raindrop model is more complicated, which can be seen as the combination of clean background B and discrete, misty raindrop D.

$$R_d = (1 - M) \odot B + D$$

(a) PreNet (b) PreNet_r (c) PRN (d)PRN_r

Figure 6. Visual images comparison of conventional ResBlocks and recursive ResBlocks.

In this model, M is a binary mask with 0 indicated that the pixel belongs to clean background and 1 represented that the pixel is rain area.

Reasonably, the test result proves that the networks in these paper cannot provide great deraining result for images with dense raindrop.

5.2 Analysis of ResBlocks

By comparing the results of different progressive networks tested on the same dataset, we can find that the performance of PreNet and PreNet_r are similar. And the test results for PRN and PRN_r cannot be distinguished visibly, which can be easily visualized in Fig. 6. Then, we printed the parameters of differnet network and show them here.

Model	PRN	PRN_r	PreNet	PreNet_r
# of parameters	95107	21,123	168,963	94,979

Table 3. Number of network parameters for different model

From the table, we can see that recursive ResBlocks deduce the number of parameters greatly but still keep great performance. Hence, PReNet_r would be better for the user who tend to derain images efficiently.

5.3 Same model training on different dataset

Also, we tried to find the difference when we using the same model to train on different dataset and test on the same dataset. Here, we conducted the experiment that trained four progressive network on three synthetic rainstreak datasets and tested on Rain12 dataset.

Training Dataset	Metric	PRN	PRN_r	PreNet	PreNet_r
TrainRainL	PSNR	36.6235	36.1576	36.6631	**36.6892**
	SSIM	0.9620	0.9605	0.9610	**0.9621**
TrainRainH	PSNR	34.3529	33.5931	**36.1539**	35.5137
	SSIM	0.9649	0.9614	**0.9690**	0.9669
TrainRain12600	PSNR	32.3536	32.5591	32.2095	**32.6241**
	SSIM	0.9205	0.9323	0.9104	**0.9195**

Table 4. Results for training same model on different dataset and test on the Rain12 dataset

As the table shows, different training dataset generated different models, which may cause huge gap between the deraining output images. When we train the network on TrainRainL dataset, it gives us the best performance on test dataset. We guessed that the patterns of images in TrainRainL and Rain12 are similar, which could be one possible reason.

6. Extension and Indivial contribution

6.1 Extension

Instead of using deeper and complex networks, PreNet applied the simple combina-

tion of ResNet and multi-stage recursion, i.e., PRN, and those methods result in a good performance on the deraining. Moreover, the deraining performance can be further boosted by the inclusion of recurrent layer, and stage-wise result is also taken as input to each ResNet, resulting in our PReNet model. Furthermore, the network parameters can be reduced by incorporating inter- and intra-stage recursive computation (PRNr and PReNetr).

Despite obvious improvements on feature representation brought by those methods, their single-scale frameworks can hardly capture the inherent correlations of rain streaks across scales. To get better performance on image deraining, people are trying to apply multi-scale learning in derain. Since rain streaks in the air show the apparent self-similarity, both within the same scale or across different scales, people are thinking about exploiting the correlated information across scales for rain streak representation.

6.2 Individual Contribution

In the experiment part , we in total apply 3 different models and use them to 6 different data sets and get the final results. To complete this project more efficiently, we divided to 2 groups to accomplish the experiments:

- Yutong Xie & Weiran Guan, response for training and testing of the PRN and PreNet models and organized the results
- Juhong Chen & Yunyi Zhang, response for training and testing RESCAN model and organized the results.

In the report part, we did some algorithm and results comparison among RESCAN, PRN and PreNet r. Additionally, we did some analysis and extension based on our experiment. For this part, we did equal work in our team.

7. Reference

[1] Ren, Dongwei & Zuo, Wangmeng & Hu, Qinghua & Zhu, Pengfei & Meng, Deyu. (2019). Progressive Image Deraining Networks: A Better and Simpler Baseline. 3932-3941. 10.1109/CVPR.2019.00406.

[2] Li, Xia & Wu, Jianlong & Lin, Zhouchen & Liu, Hong & Zha, Hongbin. (2018). Recurrent Squeeze-and-Excitation Context Aggregation Net for Single Image Deraining.

[3] Rain100H, Rain100L, RainTrainH, RainTrainL, Rain1400, Rain12, Rain12600, https://onedrive.live.com/?authkey=%21AIYIy8ZKL9kkmd4&id=66CE859AB42DFA2%2130078&cid=066CE859AB42DFA2

[4] Rain800, https://drive.google.com/drive/folders/0Bw2e6Q0nQQvGbi1xV1Yxd09rY2s

[5] AttentiveGAN-Data, https://drive.google.com/open?id=1e7R76s6vwUJxILOcAsthgDLPSnOrQ49K