Rain O'er Me: Synthesizing Real Rain to Derain With Data Distillation

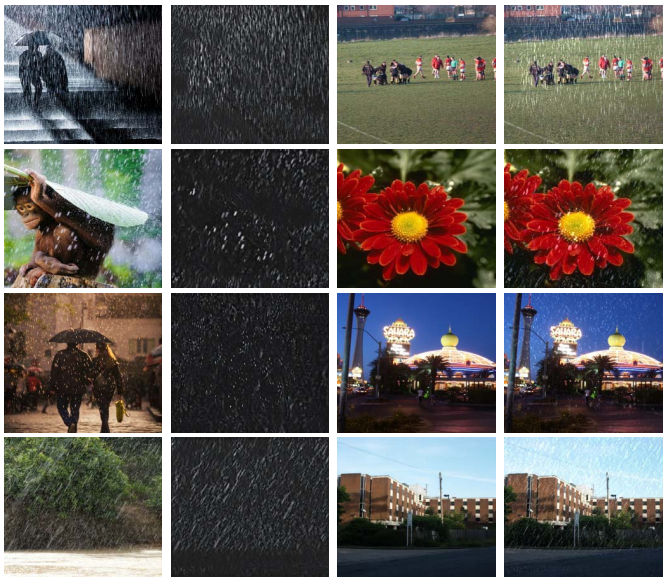
Huangxing Lin, Yanlong Li, Xueyang Fu, *Member, IEEE*, Xinghao Ding, *Member, IEEE*, Yue Huang, *Member, IEEE*, and John Paisley, *Member, IEEE*

***Abstract*— We present a weakly-supervised technique for learning to remove rain from images without using synthetic rain software. The method is based on a two-stage data distillation approach, which requires only some unpaired rainy and clean images to generate supervision. First, a rainy image is paired with a coarsely derained version using on a simple filtering technique ("rain-to-clean"). Then a clean image is randomly matched with the rainy soft-labeled pair. Through a shared deep neural network, the rain that is removed from the first image is then added to the clean image to generate a second pair ("clean-to-rain"). The neural network simultaneously learns to map both images such that high resolution structure in the clean images can inform the deraining of the rainy images. Demonstrations show that this approach can address those visual characteristics of rain not easily synthesized by software in the usual way.**

***Index Terms*— Image deraining, deep learning, unpaired data, image filter.**

I. Introduction

OUTDOOR vision systems, such as road surveillance, can be negatively impacted by rainy weather conditions [1]–[3]. Many fully-supervised convolution neural networks have been proposed to address this rain removal problem at the single-image level [4]–[8]. These methods use large number of image pairs with and without rain for training, for software is used to synthesize rain in a clean image. While performance is often good, generalization can be poor when the appearance and style of synthetic rain is different from the real rain. In Figure 2 below, we see that synthetic rain tends to be more homogeneous in shape, brightness and direction, while the distribution of real rain is much more irregular. The result is that a model trained with synthetic rain has difficulty in many realistic scenarios. Instead, adding real synthetic rain to a clean image as in Figure 1, a model can more easily learn to recognize and remove the realistic looking object.



(a) (b) (c) (d)

Fig. 1. (a) Real rainy image. (b) rain map $\hat{R}_e$ corresponding to (a) generated by our method. (c) real clean image C. (d) rainy image pair generated for training, $D = C + \hat{R}_e$.

At present, the major problem of single image rain removal is improving generalization performance. The ideal solution is to train the networks with only real-world images. Unfortunately, collecting clean/rainy versions of the exact same image is effectively impossible. Another approach is to treat rain removal as an unsupervised learning problem. Some unsupervised methods, such as CycleGAN [9] and DualGAN [10], usually use unpaired data to learn the cross-domain image translation. We note that collecting unpaired rainy and clean images is relatively easy. However, rain streaks are fairly sparse in the rainy image, these unsupervised methods tend to focus on high energy content in the absence of supervised constraints, thus failing at this problem.

To address the above problems, we focus on combining information from unpaired real rainy and clean images to mutually aid the deraining process. This is based on a two-stage data distillation method that attempts to perform deraining of both real rainy images and clean images to which the rain extracted from the rainy images has been added. Like previous knowledge distillation methods [11]–[13], our method also creates soft and hard objectives to train this single network. However, our method does not require

Manuscript received August 27, 2019; revised April 19, 2020 and June 15, 2020; accepted June 21, 2020. Date of publication July 3, 2020; date of current version July 13, 2020. The work was supported in part by the National Natural Science Foundation of China under Grants 81671766, Grant 61971369, Grant U19B2031, Grant U1605252, Grant 61671309, and Grant 61901433, in part by the Open Fund of Science and Technology on Automatic Target Recognition Laboratory under Grant 6142503190202, in part by Fundamental Research Funds for the Central Universities under Grant 20720180059, Grant 20720190116, and Grant 20720200003, and in part by the Tencent Open Fund. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Raja Bala. *(Corresponding author: Xinghao Ding.)*

Huangxing Lin, Yanlong Li, Xinghao Ding, and Yue Huang are with the School of Informatics, Xiamen University, Xiamen 361005, China (e-mail: dxh@xmu.edu.cn).

Xueyang Fu is with the School of Information Science and Technology, University of Science and Technology of China, Hefei 230026, China.

John Paisley is with the Department of Electrical Engineering, Columbia University, New York, NY 10027 USA, and also with the Data Science Institute, Columbia University, New York, NY 10027 USA.

Digital Object Identifier 10.1109/TIP.2020.3005517

Fig. 2. Visual comparisons of real rainy images and synthesized rainy images. (a)-(d): real rainy images from Internet and their corresponding rain maps generated by our deraining network. (e)-(h): synthesized rainy images and their corresponding rain maps obtained by subtracting ground-truth.

a strong pre-trained teacher network or a large amount of paired data.

Instead, we observe that rain is a form of sparse noise that can be suppressed using general image transformation techniques such as filtering (Gaussian filtering, median filtering, etc.). Therefore, it is easy to remove rain from an image with these basic techniques, but unfortunately much informative image content is removed as well due to over-smoothing. In this paper, we first use an image filter to create soft (rain-free but blurred) labels to guide the deraining process. Then, we view a deraining model as a data distillator, which can distill the noise information (*i.e.*, rain streaks) from the input rainy image to help generate new rainy-clean image pairs by adding the removed rain to a random clean image. By training a neural network on both the "soft label" (rain-to-clean) and "hard label" (clean-to-rain) images, we can simultaneously learn to preserve high resolution detail from the latter, while learning to detect and remove realistic looking rain from the former. Through experiments on several synthetic and real-world datasets, we demonstrate the effectiveness of the proposed two-stage data distillation method. In particular, our method performs well in nearly any real rainy scenario, while the robustness of other state-of-the-art deraining methods to non-uniform rain is often more disappointing.

The contributions of our paper are summarized as follows.

- We propose a new two-stage data distillation method for single image rain removal. To the best of our knowledge, this is the first method for training deraining networks using unpaired rainy and clean images.
- We construct soft and hard objectives using the content extracted from unpaired rainy and clean images. Guided by these objectives, the deraining network will restore the input rainy images to rain-free and high-quality images.
- Based on the two-stage data distillation method, we can train a model with unpaired real rainy and clean images. In this way, the deraining network is forced to learn the mapping function between real-world data, instead of synthetic data. Thus, our model can generalize well to other real-world images.

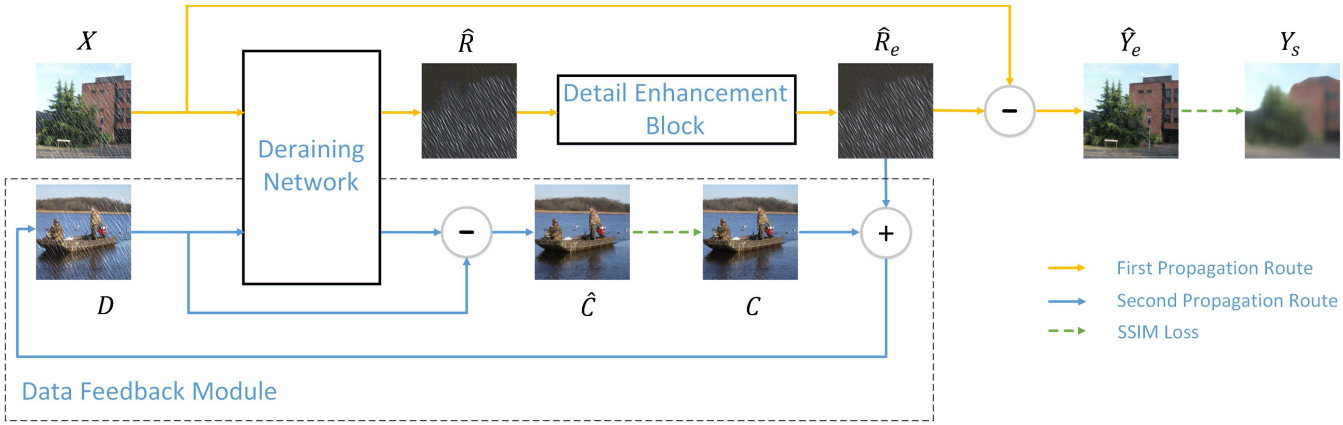
II. Related Work

1) Single Image Rain Removal: Single image deraining is a challenging and ill-posed task. Traditional methods are designed by using handcrafted image features to describe physical characteristics of rain streaks, or exploring prior knowledge to make the problem easier. For example, Kang *et al.* [14] attempt to separate rain streaks from the high frequency layer by sparse coding. Chen and Hsu [15] utilize a low-rank appearance model to capture the spatio-temporally correlated rain streaks. Kim *et al.* [16] apply a nonlocal mean filter to recover the rain streak regions. Luo *et al.* [17] propose a rain removal framework based on discriminative sparse coding. Li *et al.* [18] exploit Gaussian mixture models to separate the rain streaks. Wang *et al.* [19] combine image decomposition and dictionary learning to remove rain or snow from images. Zhu *et al.* [20] use three image priors to assist the deraining process. A limitation of many of these methods is that they tend to over-smooth the resulting image [18], [21].

Recently, deep learning has sped up the progress of single image deraining. Fu *et al.* [22] utilize domain knowledge and train a ResNet on high frequency parts to simplify the learning processing. Zhang *et al.* [23] adopt a generative adversarial network (GAN) to synthesize the derained image from a given input rainy image. To adaptively utilize the rain-density information, Zhang and Patel [6] further present a density-aware multi-stream densely connected convolutional neural network (DID-MDN) to perform the image deraining process. Yang *et al.* [24] construct a recurrent dilated network, which jointly learns three targets: rain streaks appearance, rain streaks location and derained image. To mitigate the problem of rain streak accumulation, Li *et al.* [25] develop a 2-stage CNN to remove rain streaks and rain accumulation simultaneously. Ren *et al.* [26] adopt a progressive ResNet to remove rain streaks progressively at different stages. Li *et al.* [7] combine recurrent neural networks with squeeze-and-excitation blocks [27] for rain removal. To make the trained network have better generalization ability, Wei *et al.* [28] introduce a semi-supervised transfer learning method for unseen rain types. Yasarla and Patel [29] propose an uncertainty-guided multi-scale residual learning network to learn the rain content at different scales and use them to estimate the final derained output. These methods learn a mapping between synthesized rainy images and their corresponding ground truths. A major drawback, however, is that this can lead to poor generalization ability to real rainy images that are not easily synthesized for training.

2) Knowledge Distillation: Knowledge distillation has been explored to transfer knowledge between varying-capacity networks for supervised modeling [11], [12], [30]–[32]. Hinton *et al.* [11] use the output of a large pre-trained network to aid the training of a smaller student network. Radosavovic *et al.* [13] propose data distillation, which ensembles a model run on different transformations of an unlabeled input image to improve the performance of the

(a) Coarse rain removal to generate the soft label pair.



(b) Two-stage data distillation for deraining. The network uses (a) and a random clean image pair to guide the network.

Fig. 3. The framework of our two-stage data distillation for singal image rain removal. (top) First, an image filter creates a soft label for supervision. The result is a blurred image with the rain removed. (bottom) We then use a deep network to learn how to remove the rain from the original image while preserving background details. This is achieved by pairing a rainy image with a different clean image. The deraining network then is simultaneously responsible for deraining the true image, and the clean image to which the removed rain has been added.

target model, but which cannot adapt to unsupervised tasks. Unlike [11], [13], our work focuses on distilling knowledge from the input data to construct extra supervision information without using a pre-trained teacher network.

III. Proposed Method

A rainy image X is often considered as a summation of a rain-free image Y and a rain-streak component (*i.e.*, residual map) R [20], [33]. Given X the goal of image deraining is to estimate Y, which can be equivalently done by estimating the rain residual R. Assuming that the paired data (X, Y) is not available, our goal is to train a deraining network using unpaired rainy and clean images. To generate the necessary supervision information, we propose a two-stage data distillation method as shown in Figure 3. We call this method two-stage because we distill knowledge from the input data twice to form soft and hard supervision.

- In the first stage, the rainy image passes through an image filter that easily removes the rain streaks, but along with much of the other high resolution as well. This generates a rainless "soft label" for the rainy image to help guide the deraining process.
- In the second stage, we use a deraining model as the data distillator to remove the rain from the input rainy image and add it to a different clean image to generate a new "hard labeled" image pair.

Under the guidance of the soft rain-to-clean objective, the network will learn to remove rain streaks, while the hard clean-to-rain objective will force the network to learn to output images with structural detail. Combining soft and hard tasks in one learning objective, our network learns to output high-quality rain-free images.

A. Data Transformation

Since only unpaired rainy and clean images are available, we cannot use a pre-trained teacher network to generate supervision, as is the case with previous knowledge distillation methods [11], [13], [30], [31]. Fortunately, we can use some image prior knowledge to generate extra information for learning. For example, we know that rain streaks can be significantly suppressed by some basic image transformation techniques, such as filtering. We also know that the pixel value of the rain is larger than that of the surrounding pixels.

Based on these observations, we first use an image filter to transform the unlabeled rainy image X into a soft label Y_s, *i.e.*, a rain-free but blurred image (see Figure 3(a)). The purpose is to use a fast technique for deraining that can guide the learning. Note that Y_s must be rain-free, and whether it is blurred will not affect the performance of our method. Due to the larger pixel values in the rain regions, the median filter is better at removing rain than other filtering methods (see Figure 4). Therefore, we adopt the median filter to create Y_s in experiments.

We can then pair X and Y_s to provide supervision. However, since Y_s is blurred if we use it to train the deraining network directly the results will be disappointing. To circumvent this problem, we introduce an auxiliary block into the model, called a detail enhancement block (see Figure 3(b)). To avoid confusion, we refer to the combination of the deraining network and detail enhancement block as the deraining model in this paper. We provide more details about it in the following subsections.

Given a rainy input X, the direct output of the deraining model is a residual map $\hat{R}_e$, and the corresponding rain

Fig. 4. One example of data transformation using different methods. (a) Rainy image X. (b) Gaussian filter, kernel size=31. (c) Mean filter, kernel size = 31. (d) Median filter, kernel size = 21.(e) Median filter, kernel size = 31. (f) Median filter, kernel size = 41.

removal result is obtained by

$$\hat{Y}_e = X - \hat{R}_e. \tag{1}$$

The soft objective for the deraining process has the following loss function

$$L_S = \frac{1}{M}\sum_{i=1}^{M} 1 - SSIM(\hat{Y}_e^i, Y_s^i), \tag{2}$$

where $SSIM$ represents the SSIM loss [34] and M is the number of training data.

B. Data Feedback

Following the soft objective Eq. (2), $\hat{Y}_e$ will be rain-free but blurred. To obtain high-quality derained results, some additional constraints should be introduced so that the network can learn to preserve the image details while removing rain.

We find that the deraining model can be regarded as a data distillator. The process of the model taking in the input image X and generating the corresponding residual map $\hat{R}_e$ can be seen as a distillation of X. The residual map $\hat{R}_e$ contains all the rain streaks of the input rainy image X, and we can use this to generate extra knowledge. Specifically, if we add $\hat{R}_e$ to another clean image C to generate a new rainy image D,

$$D = C + \hat{R}_e. \tag{3}$$

Then we have a new rainy/clean pair (*i.e.*, D and C). The new rainy image D and input image X have the same rain streaks, but their backgrounds are different; both are sharp, but the soft label for X is more blurry while the rainy D is still high resolution. We refer to the new image pair C and D as a hard labeled image pair because the rainy image D has a corresponding clean and sharp label C. We then use the hard labeled image pair to provide a hard objective for deraining. However, soft and hard objectives are competing and they cannot work in tandem for the same network. To mitigate this problem, we skip the detail enhancement block and only apply the hard objective to the deraining network (see Figure 3(b)). The deraining network is therefore guided to retain image details while removing rain by the following hard objective,

TABLE I

EFFECTS OF L_H AND L_S ON $\hat{R}$ AND $\hat{R}_e$

	$\hat{R} = R + \hat{\epsilon}$		$\hat{R}_e = R + \hat{\epsilon}_e$	
	R	$\hat{\epsilon}$	R	$\hat{\epsilon}_e$
L_H	Maintain	Decay	-	-
L_S	Maintain	Enhance	Maintain	Enhance

$$L_H = \frac{1}{M}\sum_{i=1}^{M} 1 - SSIM(\hat{C}^i, C^i), \tag{4}$$

$$\hat{C} = D - f(D; \theta_f), \tag{5}$$

where f denotes the mapping function of the deraining network, θ_f represents parameters of the network. Eq. (5) implies that the direct output of the deraining network is also a residual map. Combining the hard objective Eq. (4) with the soft objective Eq. (2), the complete objective function L is

$$L = \alpha \cdot L_H + \beta \cdot L_S, \tag{6}$$

where α and β represent the weights of the two objectives.

Note that, the input to the detail enhancement block is a residual map $\hat{R}$, which is output by the deraining network,

$$\begin{aligned} \hat{R} &= f(X; \theta_f) \\ &= R + \hat{\epsilon}, \end{aligned} \tag{7}$$

where $\hat{\epsilon}$ represents some image details from the input rainy image X. Guided by the soft objective Eq. (2), the detail enhancement block learns to transform $\hat{R}$ into a new residual map $\hat{R}_e$ with richer image details,

$$\begin{aligned} \hat{R}_e &= g(\hat{R}; \theta_g) \\ &= R + \hat{\epsilon}_e, \end{aligned} \tag{8}$$

where g denotes the mapping function of the detail enhancement block, θ_g represents parameters of this block, and $\hat{\epsilon}_e$ is an enhanced version of $\hat{\epsilon}$, which is used to blur the resulting image by Eq. (1). This is why we call this block the detail enhancement block.

On the other hand, the hard objective L_H encourages the deraining network to preserve image details while removing rain. This means that the $\hat{\epsilon}$ in $\hat{R}$ Eq. (7) will gradually decay. Ideally, a deraining network should be able to maintain only R in $\hat{R}$ but reduce $\|\hat{\epsilon}\|$ to 0. However, based on Eq. (6), the soft objective L_S also has an effect on the deraining network due to gradient backpropagation. L_S and L_H are adversarial, the soft objective L_S will not allow $\|\hat{\epsilon}\|$ to converge to 0 (see Table I). Because if $\|\hat{\epsilon}\| = 0$, the soft objective will be an impossible task. Following Eq. (8), the detail enhancement block can always keep the rain streaks R in $\hat{R}_e$, but it is impossible to transform 0 into the specific image details. Therefore, the deraining network has to make a trade-off between these two objectives, which results in $\|\hat{\epsilon}\|$ should be greater than 0. Only in this case can the detail enhancement block reduce L_S by Eq. (8). One possible way to weaken the impact of the soft

Algorithm 1 Two-Stage Data Distillation for Image Deraining

Require: Unpaired rainy and clean datasets. An image filter.

1: **for** $t = 1$ **to** T **do**
2: Sample minibatch of M rainy images $\{X^1, ..., X^M\}$ from the rainy dataset.
3: Input $\{X^1, ..., X^M\}$ to an image filter, and generate soft labels $\{Y_s^1, ..., Y_s^M\}$.
4: Input $\{X^1, ..., X^M\}$ to deraining network, and generate $\{\hat{R}^1, ..., \hat{R}^M\}$ with Eq. (7).
5: Input $\{\hat{R}^1, ..., \hat{R}^M\}$ to detail enhancement block, and generate $\{\hat{R}_e^1, ..., \hat{R}_e^M\}$ with Eq. (8).
6: Sample minibatch of M clean images $\{C^1, ..., C^M\}$ from the clean dataset.
7: Add $\{\hat{R}_e^1, ..., \hat{R}_e^M\}$ to $\{C^1, ..., C^M\}$ to get new rainy images $\{D^1, ..., D^M\}$.
8: Input $\{D^1, ..., D^M\}$ to deraining network, and generate $\{\hat{C}^1, ..., \hat{C}^M\}$ with Eq. (5).
9: Update α and β with Eqs. (9) and (10).
10: Update the deraining network and detail enhancement block with Eq. (6).
11: **end for**

objective on the deraining network is to set α in Eq. (6) to a large value and β to a small value (*e.g.* $\alpha = 100, \beta = 1$). However, this approach has two flaws. First, if α and β are both fixed values, no matter how large α is, the deraining network will converge to a bad equilibrium between soft and hard objectives, where $\|\hat{\epsilon}\|$ is greater than 0. Second, a large α will lead to a large gradient for the deraining network at the beginning of training, making it difficult to converge.

To solve the above problems, instead of setting α and β in Eq. (6) to fixed values, we change them dynamically,

$$\alpha = \frac{t}{1000}, \tag{9}$$

$$\beta = \frac{1000}{1000 + t}, \tag{10}$$

where t represents the number of iterations in the training. (See Algorithm 1.)

From Eqs. (9) and (10), we see that α gradually increases from 0 and β decreases from 1 during training. By dynamically adjusting α and β, the balance between soft and hard obejectives is constantly disrupted. As α gets larger, the deraining network will only need to focus on hard objective, while the soft objective will be negligible. Therefore, the deraining network learns to preserve image details while removing rain, which means the $\hat{\epsilon}$ in $\hat{R}$ Eq. (7) will guadually decay to 0. The weakening of $\hat{\epsilon}$ will also reduce the image details (*i.e.*, $\hat{\epsilon}_e$) contained in $\hat{R}_e$ (see Figure 5). After convergence, $\|\hat{\epsilon}\| \approx 0$, *i.e.*,

$$\hat{R} = f(X; \theta_f) \approx R. \tag{11}$$

Following Eqs. (11) and (8), the detail enhancement block can only maintain the R in $\hat{R}_e$, it is almost impossible to generate additional image details (*i.e.* $\|\hat{\epsilon}_e\| \approx 0$). Therefore,

$$\hat{R}_e = g(\hat{R}; \theta_g) \approx \hat{R} \approx R. \tag{12}$$

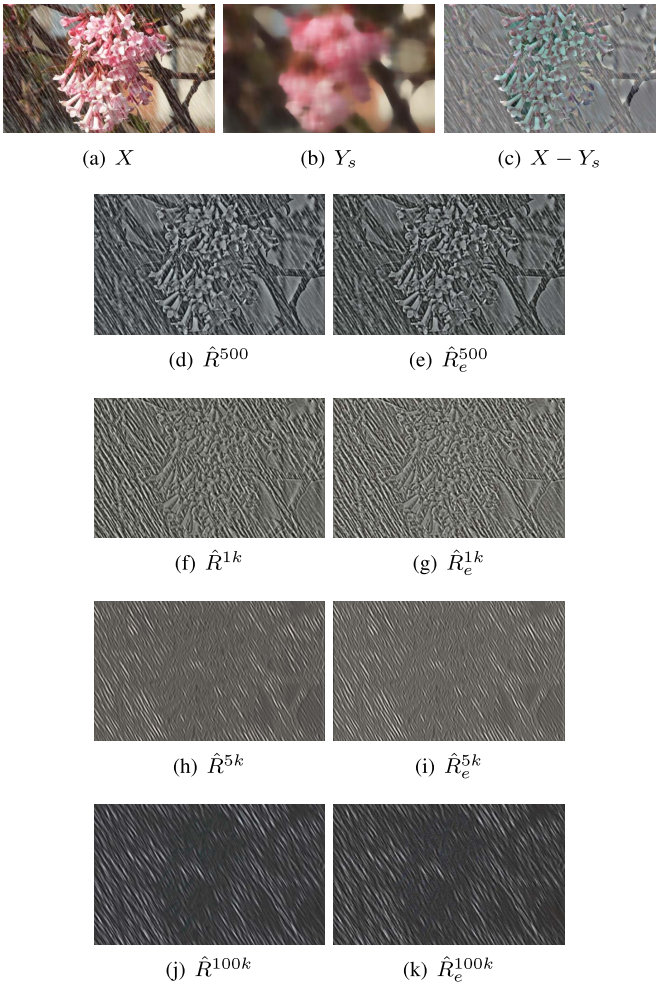


Fig. 5. Illustrations of the convergence process of our deraining model. (a)-(c): Rainy image X, soft label Y_s and a reidual map obtained by $X - Y_s$. (d)-(k): The deraining network gradually learns to preserve image details while removing rain, so the image details $\hat{\epsilon}$ in $\hat{R}$ gradually decay to 0. The weakening of $\hat{\epsilon}$ will also reduce the image details contained in $\hat{R}_e$. After convergence, $\hat{R}_e$ will consist of rain streaks only. The superscript represents the number of iterations in training.

Note that when testing, we remove the detail enhancement block and use the deraining network directly to get the final output $\hat{Y}$,

$$\hat{Y} = X - f(X; \theta_f). \tag{13}$$

By the way, Eq. (12) implies that after convergence, the output of the detail enhancement block is almost the same as that of the deraining network (*i.e.* $\hat{Y} \approx \hat{Y}_e$).

C. Network Architecture

Our model consists of two parts, a deraining network and a detail enhancement block. Here, we introduce the network architecture of each part. The architecture of the detail enhancement block has little effect on our method, so we adopt a simple 3-layer convolutional network as the detail enhancement block.

For deraining network, many well-designed network architectures [5]–[7], [22] have been proposed to better remove rain. Most existing deep methods design a very complex network

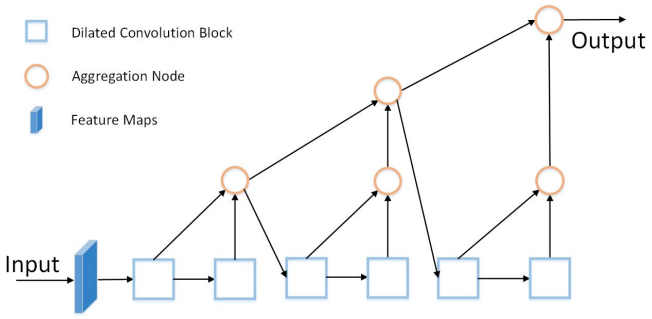


Fig. 6. The hierarchical aggregation architecture of our deraining network. Hierarchical aggregation learns to extract the full spectrum of semantic and spatial information from the network. The deraining network contains six dilated blocks.

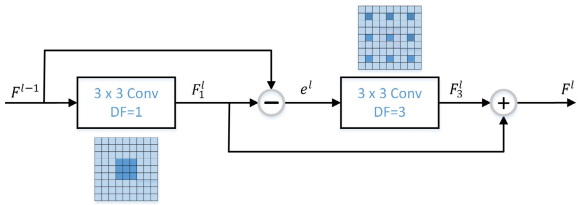


Fig. 7. The structure of the dilated convolution block. DF indicates the dilated factor.

in order to obtain higher numerical performance on synthetic datasets, but at the cost of some poor generalization, scalability and practicality in real-world image applications.

Instead of directly cascading convolutional layers, we design a hierarchical aggregation architecture, as shown in Figure 6, to better fuse spatial and semantic information across layers [35], which can lead to high quality images reconstruction. We argue that effectively aggregating features can lead to better rain removal as well as better parameter and memory efficiency. On the other hand, unlike the usual noise, the appearance of the rain streak is irregular, as shown in Figure 2. To better capture the characteristics of rain, we design the multi-scale dilated convolution block shown in Figure 7 as the backbone of the deraining network. The block is defined as follows,

$$\begin{aligned} F_1^l &= W_1^l * F^{l-1}, \\ e^l &= F^{l-1} - F_1^l, \\ F_3^l &= W_3^l * e^l, \\ F^l &= F_1^l + F_3^l, \end{aligned} \tag{14}$$

where F and W are feature maps and 3×3 convolution kernels, respectively. The subscript number are dilation factors, $*$ indicates the convolution operation, l indexes block number. The multi-scale dilated convolution block can also be viewed as a self-correcting procedure that feeds a mapping error to the sampling layer at another scale and iteratively corrects the solution by feeding back the mapping error. Moreover, the features between blocks are fused by the aggregation nodes. The nodes learn to select and project important information to maintain the same dimension at their output as a single input.

For both the deraining network and detail enhancement block, the kernel size of each convolutional layer is 3×3, and the number of feature maps is 16. The activation function for all convolutional layers is $ReLU$ [36], while the activation function of the last layer is $Tanh$.

D. Datasets

It is hard to obtain rainy/clean image pairs from real-world data, but it is relatively easy to collect a large number of real rainy images. We collect 2400 real rainy images from the Internet, which are diverse in background and rain, to form a new rainy dataset called $RealRain2400$.[1] We divide these images into a training set and a testing set in a radio of 7:1. In our method, we also need some clean images to generate supervision. We use the 4744 natural images provided by [37] as the clean set, and refer to it $Clean4744$. Note that the images in $Clean4744$ are not related to those in $RealRain2400$.

E. Training Details and Parameter Settings

We use Pytorch and Adam [38] with a mini-batch size of 8 to train our model. For all experiments, we use a median filter with a kernel size of 31 to generate soft labels. We randomly select 128×128 image patchs from training set as inputs. We set the learning rate as 1×10^{-4} for the first 150K iterations and linearly decay the rate to 0 for the next 150K iterations. All experiments are performed on a server with Inter Core i7-8700K CPU and NVIDIA GeForce GTX 1080 Ti.

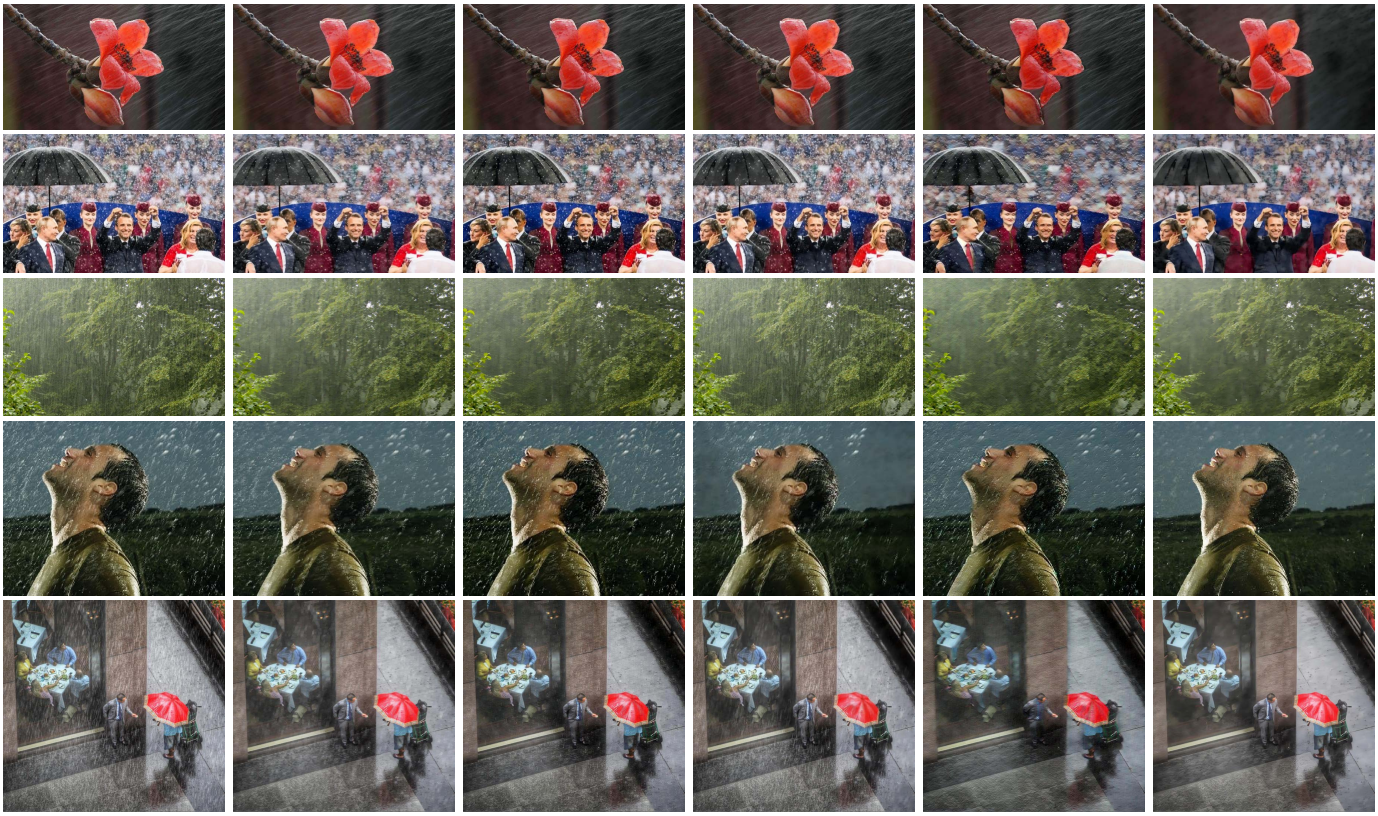
IV. Experiments

We compare our method with several state-of-the-art deraining methods: Joint Convolutional Analysis and Synthesis (JCAS) [39], DDN [22], RESCAN [7], DID-MDN [6], JORDER-E [24], SPANet [40], SS-IRR [28], UMRL [29] and PReNet [26]. Unlike previous methods, which only pursue higher numerical metrics on *synthetic* data, our desire is devoted to a better qualitative generalization to real-world scenarios.

A. Real-World Data

We use the testing set provided by SPANet [40] to conduct a quantitative comparison. This testing set, which we call $Test1000$, contains 1000 pairs of images with a size of 512×512. The rainy images and their corresponding ground-truth in $Test1000$ are extracted from video, so they can be regarded as real-wolrd data. Our model is trained with $RealRain2400$ and $Clean4744$. The pre-trained models of the compared deep methods (except RESCAN) have been released online, and we test them directly on $Test1000$. Since all methods, except SPANet [40], have not trained on the data from [40], the comparison on $Test1000$ can well reflect the generalization performance of each method. SSIM and PSNR are adopted to perform quantitative evaluations in Table II. It is not surprising that SPANet achieves the highest metrics. Besides that, our method significantly outperforms other deep methods. This experiment shows that our method

[1] We will release our code and data.



(a) Rainy images (b) JCAS (c) DDN (d) SPANet (e) JORDER-E (f) Ours

Fig. 8. Visual comparisons on real rainy images. We have chosen images that do not have typical look of synthetic rain and which would be difficult to generate using current software. In many cases, the existing algorithms can remove rain that is homogenous and synthetic-like. However, when this is not the case our algorithm still removes the rain while others fail. This is because data distillation allows our network to be trained on rain with this appearance, while the other algorithms are not.

TABLE II

QUANTITATIVE EVALUATION RESULTS ON *Test*1000 DATASET

	JCAS	DDN	DID-MDN	UMRL	JORDER-E	PReNet	SS-IRR	SPANet	Ours
SSIM	0.937	0.928	0.815	0.866	0.923	0.934	0.894	**0.967**	0.953
PSNR	33.61	32.54	22.38	26.59	32.14	33.21	28.48	**35.76**	34.66

can generalize to real rain better than other CNNs trained on synthetic data.

We show the qualitative results on real-world rainy images in Figures 8 and 9. As can be seen, the model-based method JCAS fails to remove rain due to the modeling limitations. The effects of other fully-supervised methods are disappointing since they cannot remove the real rain that they haven't seen in their paired training data. On the other hand, our method can remove many types of rain, from small raindrops to long rain streaks, and reconstruct an image that still preserves details. We argue that, compared with other methods this approach is more robust to realistic data.

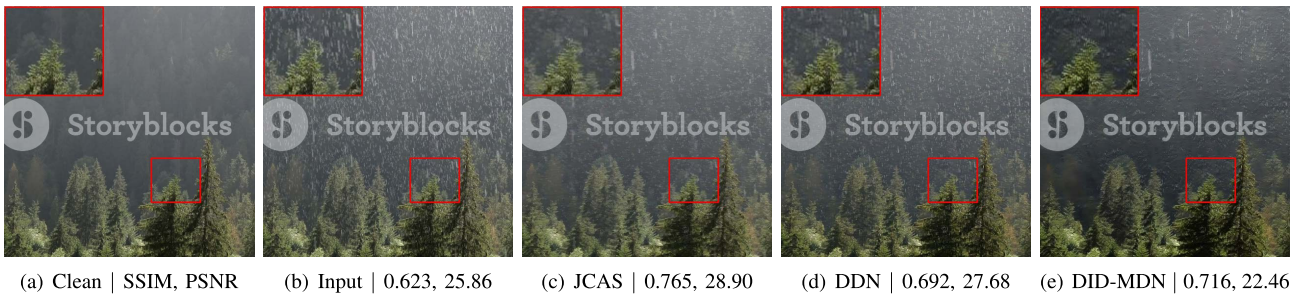
TABLE III

QUANTITATIVE RESULTS ON SYNTHETIC DATASETS

Datasets	*Rain200L*		*DDN-data*		*DID-data*	
Methods	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR
JCAS	0.878	28.32	0.822	25.49	0.781	23.84
DDN	0.926	31.75	0.872	28.58	0.865	29.07
DID-MDN	0.939	33.16	0.900	29.46	0.889	29.15
RESCAN	0.968	35.22	0.908	30.60	0.897	31.80
UMRL	0.957	33.83	0.905	29.69	0.901	30.99
JORDER-E	**0.978**	**35.87**	**0.929**	**31.53**	**0.912**	**32.11**
PReNet	0.974	35.66	0.925	31.28	0.910	31.82
SPANet	0.962	32.88	0.902	30.04	0.905	31.39
SS-IRR	0.938	32.29	0.875	28.62	0.868	29.12
Ours-sv	0.970	35.37	0.915	30.49	0.902	30.86
Ours	0.955	33.09	0.890	28.50	0.873	28.72

B. Synthetic Data

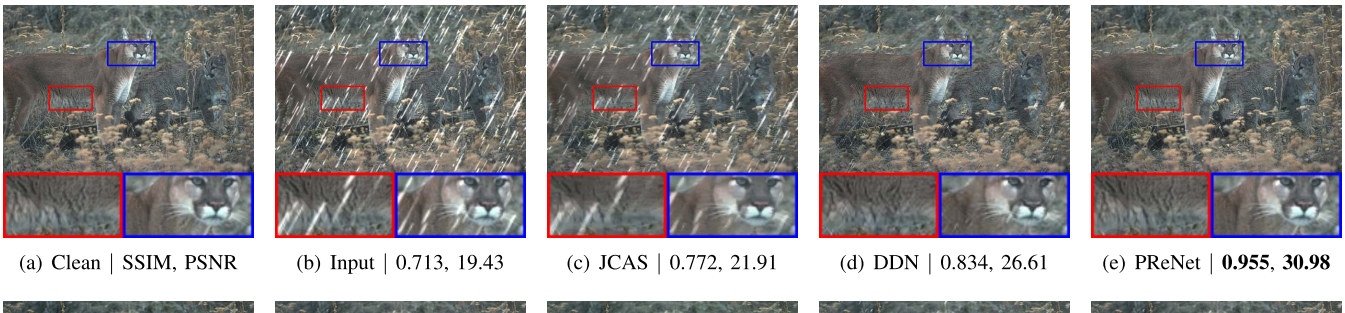
The appearance and distribution of synthetic rain are very different from the real rain. However, synthetic images are perfectly aligned and high quality images, which can be used to further evaluate the effectiveness of our method. We use three public synthetic datasets provided by JORDER-E [24], DDN [22] and DID-MDN [6] for comparison. The rain streaks in these three datasets were synthesized using different strategies. These three datasets contain 200, 1400 and 1200 testing images, respectively. We call them *Rain200L*, *DDN-data* and *DID-data*. For a fair comparison, all deep methods are retrained with these synthetic data. Note that, using our two-stage data distillation approach, training the deraining network requires only the synthetic rainy images,



(a) Clean | SSIM, PSNR (b) Input | 0.623, 25.86 (c) JCAS | 0.765, 28.90 (d) DDN | 0.692, 27.68 (e) DID-MDN | 0.716, 22.46

(f) UMRL | 0.767, 27.22 (g) JORDER-E|0.794, 25.88 (h) SS-IRR | 0.713, 26.97 (i) SPANet | **0.921**, 32.66 (j) Ours | 0.919, **32.88**

Fig. 9. One visual comparison on the real-world dataset "$Test1000$."



(a) Clean | SSIM, PSNR (b) Input | 0.713, 19.43 (c) JCAS | 0.772, 21.91 (d) DDN | 0.834, 26.61 (e) PReNet | **0.955**, **30.98**

(f) UMRL | 0.928, 27.20 (g) JORDER-E|0.947, 30.42 (h) SPANet | 0.921, 29.30 (i) Ours | 0.927, 29.14 (j) Ours-sv | 0.944, 30.53

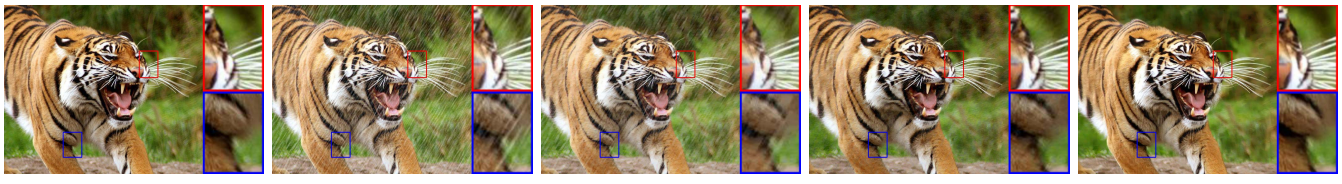
Fig. 10. One visual comparison on the synthetic dataset "*Rain200L*."

not ground-truth. We again use $Clean4744$ to generate the hard objective for our model. To evaluate the gap between our approach and the fully supervised methods, we also train our deraining network directly in a fully supervised ("sv") manner. We call this trained model "Ours-sv".

We show the quantitative evaluation in Table III. Admittedly, our two-stage data distillation method currently cannot outperform other fully-supervised methods on synthetic datasets, but the results are satisfactory. In many real-world cases, fully-supervised methods are often impractical, due to the lack of paired data. Our method can still achieve decent results in the absence of paired data, which shows the potential value of our method in practical application. We also show the visual results in Figures 10 to 12. As can be seen, our deraining network can achieve comparable deraining performance with other fully-supervised methods and the resulting images are sharp and clean.

C. Running Time

The specific deraining network we use can also process new images very efficiently. Table IV shows the average running time of 100 testing images, all the test are conducted with a 512×512 rainy image as input. The JCAS is a non-deep method that is run on CPU according to the provided code, while other deep methods are tested on both CPU and GPU. Compared with other methods, our network has a relatively fast speed on both CPU and GPU. As a pre-processing for other high-level vision tasks, the rain streaks removal process should be simpler and faster. Our deraining network is a



(a) Clean | SSIM, PSNR (b) Input | 0.730, 21.72 (c) JCAS | 0.800, 23.27 (d) DDN | 0.863, 28.52 (e) PReNet | 0.919, 30.83

(f) UMRL | 0.910, 30.76 (g) JORDER-E| **0.922**, **31.53** (h) SPANet | 0.880, 27.58 (i) Ours | 0.894, 28.36 (j) Ours-sv | 0.907, 30.15

Fig. 11. One visual comparison on synthetic dataset "*DDN-data*."

(a) Clean | SSIM, PSNR (b) Input | 0.725, 20.89 (c) JCAS | 0.806, 22.82 (d) DDN | 0.864, 29.15 (e) PReNet | 0.898, 30.12

(f) UMRL | 0.900, **30.75** (g) JORDER-E| **0.904**, 30.70 (h) SPANet | 0.899, 29.75 (i) Ours | 0.875, 29.40 (j) Ours-sv | 0.892, 29.79

Fig. 12. One visual comparison on the synthetic dataset "*DID-data*."

(a) Clean | SSIM, PSNR (b) Input | 0.599, 19.91 (c) $\hat{Y}$ | **0.875, 28.56** (d) $\hat{Y}_e$ | 0.863, 28.23

Fig. 13. Deraining results of our deraining nertwork and detail enhancement block.

TABLE IV

COMPARISON OF PARAMETERS AND RUNNING TIME (SECONDS). THE SIZE OF THE TESTING IMAGES: 512×512

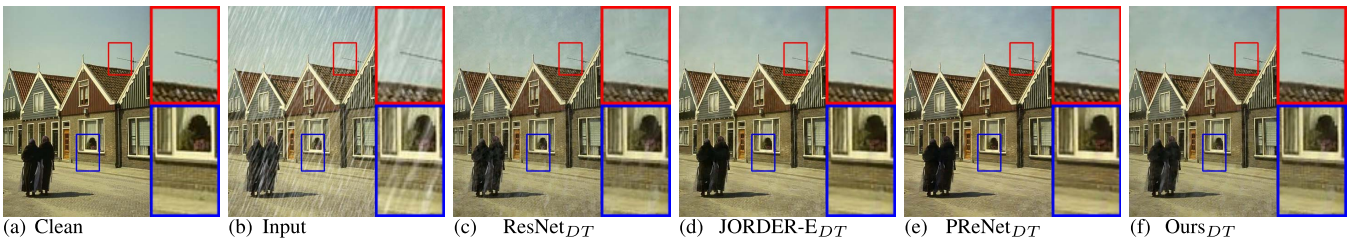
	JCAS	DDN	DID-MDN	RESCAN	UMRL	JORDER-E	PReNet	SPANet	SS-IRR	Ours
CPU	187.89	1.56	5.73	4.62	15.50	55.21	24.66	8.53	1.92	1.45
GPU	-	0.13	0.20	0.18	0.30	0.38	0.25	0.19	0.15	0.13
Parameters #	-	58, 175	135, 800	54, 735	984, 356	4, 169, 024	168, 963	283, 716	57, 369	51, 552

relatively shallow network that requires fewer calculations, so it is more practical, for example, on mobile devices.

D. Ablation Study

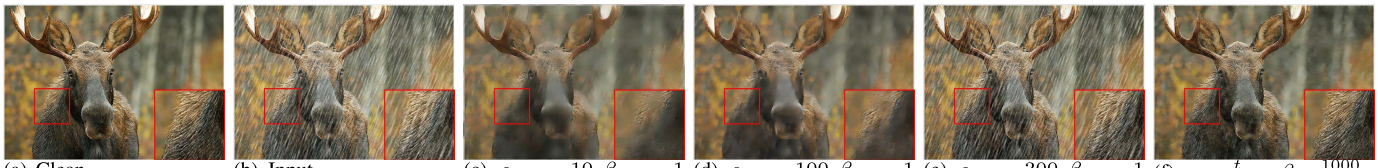
We provide ablation studies to explore the effect of each part of our model over the *DDN-data* [22].

1) Output of the Detail Enhancement Block: our model consists of two parts, a deraining network and a detail enhancement block. The input of the detail enhancement block is a residual map $\hat{R}$ generated by the deraining network. After the training has converged, the deraining network can preserve image details while removing rain. This results in almost no image details remaining in $\hat{R}$, making the detail enhancement block unable to blur the derained image. As a result, the output of the detail enhancement block (*i.e.*, $\hat{Y}_e$) is almost the same as the output of the deraining network



(a) Clean SSIM, PSNR | (b) Input 0.701, 20.38 | (c) ResNet$_{DT}$ 0.879, 26.57 | (d) JORDER-E$_{DT}$ **0.915**, 28.37 | (e) PReNet$_{DT}$ 0.910, **28.38** | (f) Ours$_{DT}$ 0.908, 27.73

Fig. 14. Deraining results by using different deraining nertworks. The subscript "DT" indicates that these networks are trained using our two-stage data distillation method.



(a) Clean SSIM, PSNR | (b) Input 0.576, 19.66 | (c) $\alpha = 10, \beta = 1$ 0.716, 22.78 | (d) $\alpha = 100, \beta = 1$ 0.755, 24.39 | (e) $\alpha = 300, \beta = 1$ 0.599, 22.16 | (f) $\alpha=\frac{t}{1000}$, $\beta=\frac{1000}{1000+t}$ **0.895, 28.97**

Fig. 15. Deraining results by using different α and β.

TABLE V

QUANTITATIVE RESULTS USING DIFFERENT MEIDAN FILTERS

Kernel size	21	31	41	51
SSIM	0.882	0.890	0.892	0.881
PSNR	28.47	28.50	28.35	28.55

TABLE VI

QUANTITATIVE RESULTS USING DIFFERENT NETWORK ARCHITECTURES. (SV: FULLY SUPERVISED. DT: TWO-STAGE DATA DISTILLATION)

	SV		*DT*	
	SSIM	PSNR	SSIM	PSNR
ResNet	0.870	28.22	0.861	27.05
JORDER-E	0.929	31.53	0.900	29.26
PReNet	0.925	31.28	0.897	29.13
Ours	0.915	30.49	0.890	28.50

(*i.e.*, $\hat{Y}$). For the *DDN-data*, the detail enhancement block yields results 0.880/28.34 dB in terms of SSIM and PSNR, close to the deraining network of 0.890/28.50 dB. Subjective results are presented in Figure 13. The deraining network and the detail enhancement block show similar deraining performance.

2) Effect of Soft Labels: in the above experiments, we use a median filter with a kernel size of 31 to create soft labels. We further explore the impact of soft labels on our method. We use median filters with different kernel sizes $\{21, 31, 41, 51\}$ to generate different soft labels (see Figure 4). The larger the kernel size, the more blurred the soft label. Quantitative results are shown in Table V. As can be seen, our method achieves similar results using different soft labels. This means that our method is not sensitive to the quality of soft labels. The only requirement is that the soft labels must be rain-free.

3) Effect of Network Architectures: we also test the impact of deraining networks with different architectures on our method. We compare our hierarchical aggregation network with 3 networks: a simple ResNet, which is the same as the backbone of DDN; JORDER-E; PReNet. For JORDER-E and PReNet, we use them as the deraining network in our method and use their default loss function to replace the hard objective. Quantitative results are shown in Table VI. We observe that our two-stage data distillation method is as sensitive as the supervised method to the architecture of the deraining network. This means that we can further improve the deraining results of our method by adopting a better network architecture. Note that although the deraining performance of our hierarchical aggregation network is inferior to JORDER-E and PReNet, the computing resources required by JORDER-E and PReNet greatly exceed our network (see Table IV). To balance performance and computational efficiency, our hierarchical aggregation network is used as the default deraining network in our method. Visual comparisons are shown in Figure 14.

TABLE VII

QUANTITATIVE RESULTS USING DIFFERENT α AND β

	α=10, β =1	α=100, β =1	α=300, β =1	$\alpha=\frac{t}{1000}$, $\beta=\frac{1000}{1000+t}$
SSIM	0.701	0.725	0.771	**0.890**
PSNR	22.78	23.12	25.05	**28.50**

4) Effect of α and β: in Eq. (6), we use α and β to weight the hard and soft objectives. We dynamically change the values of α and β during training. In this experiment, we further explore the impact of α and β, which are set to fixed values. We test $\beta = 1$ and α is selected from $\{10, 100, 300\}$. As shown in Table VII, when α and β are set to fixed values, the performance of our method is disappointing. No matter how large α is, the deraining network has to make a tradeoff between hard and soft objectives, so it fails to produce high-quality rain-free images. If we dynamically adjust α and β, the balance between hard and soft objectives will be constantly disrupted. As α increases and β decreases, the deraining network will only need to focus on the hard objective, *i.e.* removing rain and preserving image details. Subjective results are shown in Figure 15. We see that if α is set to 10 or 100, the deraining network tends to lose some image details while removing rain. We also notice that the deraining network

TABLE VIII
QUANTITATIVE RESULTS USING DIFFERENT NUMBER OF DILATED BLOCKS

Block number	4	6 (default)	8	10
SSIM	0.875	0.890	0.892	0.895
PSNR	27.38	28.50	28.65	28.71

(a) Real snow images (b) Our results

Fig. 16. Extension on single image snow removal.

fails to remove the rain when $\alpha = 300$. This is because a large α will lead to a large gradient for the deraining network at the beginning of training, making it difficult to converge. In contrast, by dynamically adjusting α and β, the deraining network successfully learns to remove rain, preserve image details, and improve visual quality.

5) Number of Dilated Blocks: Our hierarchical aggregation network consists of six dilated blocks, and we also test the impact of the block numbers. We choose the block number from the set $\{4, 6, 8, 10\}$. Quantitative results are shown in Table VIII. As can be seen, increasing the block number can generate better results due to larger modeling capacity. However, larger block number has a limited contribution to performance at the expense of computing speed and storage. To make a trade-off between performance and speed, we set the number of dilated blocks to 6.

E. Extensions

Our deraining network trained with $RealRain2400$ can be applied directly to image snow removal, as shown in Figure 16. This is because the appearance and distribution of snow is similar to that of some types of rain. Based on this observation, we can further infer that our two-stage data distillation method can also be applied to other image reconstruction tasks, such as denoising and image inpainting, under specific parameter settings. These tasks all seek to restore a clean image from the damaged input image, which is damaged in a manner similar to that of some rainy images.

On the other hand, we extend our mission to semantic segmentation to verify the potential value of our network in practical applications. Since rain streaks can blur and block objects, the performance of semantic segmentation will degrade in rainy weather. Figure 17 shows visual results of

(a) Rainy images (b) DID-MDN (c) DDN (d) Ours

Fig. 17. The 1st and 3rd rows: real rainy images and rain removal results generated by DID-MDN, DDN and our deraining network. The 2nd row: the corresponding segmentation maps of the 1st row generated using DeepLabv3+ [41]. The 4th row: the corresponding segmentation maps of the 3rd row.

semantic segmentation by combining with DeepLabv3+ [41]. It is obviously that rain streaks can degrade the performance of DeepLabv3+, *i.e.*, by missing objects and producing poor segmentation maps. Compared with other methods, our method can remove rain streaks more effectively and deliver better segmentation results along object boundaries.

V. CONCLUSION

We have proposed a two-stage data distillation method for single image rain removal. Instead of using a large amount of paired synthetic data to train a non-robust network, we focus on training a deraining network with powerful generalization capabilities using only real rainy images. In the absence of a clean label, we distill knowledge from the input data twice to construct the corresponding soft and hard objectives. Guided by the soft and hard objectives, our deraining network learns to map the input rainy image into a high-quality rain-free image by transferring rain to a high-quality clean image to create a more realistic training pair. Experiments verify the superiority of our two-stage data distillation method on real data and also shows the potential of our method to other vision and image restoration tasks.

REFERENCES

[1] W. Wei, L. Yi, Q. Xie, Q. Zhao, D. Meng, and Z. Xu, "Should we encode rain streaks in video as deterministic or stochastic?" in *Proc. ICCV*, 2017, pp. 2516–2525.

[2] M. Li *et al.*, "Video rain streak removal by multiscale convolutional sparse coding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6644–6653.

[3] H. Zhang and V. M. Patel, "Convolutional sparse and low-rank coding-based rain streak removal," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 1259–1267.

[4] W. Yang, R. T. Tan, S. Wang, Y. Fang, and J. Liu, "Single image deraining: From model-based to data-driven and beyond," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, May 19, 2020, doi: 10.1109/TPAMI.2020.2995190.

[5] W. Yang, R. T. Tan, J. Feng, J. Liu, Z. Guo, and S. Yan, "Deep joint rain detection and removal from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1357–1366.

[6] H. Zhang and V. M. Patel, "Density-aware single image de-raining using a multi-stream dense network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 695–704.

[7] X. Li, J. Wu, Z. Lin, H. Liu, and H. Zha, "Recurrent squeeze-and-excitation context aggregation net for single image deraining," in *Proc. ECCV*, 2018, pp. 254–269.

[8] D. Eigen, D. Krishnan, and R. Fergus, "Restoring an image taken through a window covered with dirt or rain," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 633–640.

[9] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2223–2232

[10] Z. Yi, H. Zhang, P. Tan, and M. Gong, "DualGAN: Unsupervised dual learning for image-to-image translation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2849–2857.

[11] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *Comput. Sci.*, vol. 14, no. 7, pp. 38–39, 2015.

[12] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4320–4328.

[13] I. Radosavovic, P. Dollar, R. Girshick, G. Gkioxari, and K. He, "Data distillation: Towards omni-supervised learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4119–4128.

[14] L.-W. Kang, C.-W. Lin, and Y.-H. Fu, "Automatic single-image-based rain streaks removal via image decomposition," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1742–1755, Apr. 2012.

[15] Y.-L. Chen and C.-T. Hsu, "A generalized low-rank appearance model for spatio-temporally correlated rain streaks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1968–1975.

[16] J.-H. Kim, C. Lee, J.-Y. Sim, and C.-S. Kim, "Single-image deraining using an adaptive nonlocal means filter," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2013, pp. 914–917.

[17] Y. Luo, Y. Xu, and H. Ji, "Removing rain from a single image via discriminative sparse coding," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3397–3405.

[18] Y. Li, R. T. Tan, X. Guo, J. Lu, and M. S. Brown, "Rain streak removal using layer priors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2736–2744.

[19] Y. Wang, S. Liu, C. Chen, and B. Zeng, "A hierarchical approach for rain or snow removing in a single color image," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 3936–3950, Aug. 2017.

[20] L. Zhu, C.-W. Fu, D. Lischinski, and P.-A. Heng, "Joint bi-layer optimization for single-image rain streak removal," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2526–2534.

[21] Y. Chang, L. Yan, and S. Zhong, "Transformed low-rank model for line pattern noise removal," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1726–1734.

[22] X. Fu, J. Huang, D. Zeng, Y. Huang, X. Ding, and J. Paisley, "Removing rain from single images via a deep detail network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3855–3863.

[23] H. Zhang, V. Sindagi, and V. M. Patel, "Image de-raining using a conditional generative adversarial network," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Jun. 3, 2019, doi: 10.1109/TCSVT.2019.2920407.

[24] W. Yang, R. T. Tan, J. Feng, Z. Guo, S. Yan, and J. Liu, "Joint rain detection and removal from a single image with contextualized deep networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 6, pp. 1377–1393, Jun. 2020.

[25] R. Li, L.-F. Cheong, and R. T. Tan, "Heavy rain image restoration: Integrating physics model and conditional adversarial learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1633–1642.

[26] D. Ren, W. Zuo, Q. Hu, P. Zhu, and D. Meng, "Progressive image deraining networks: A better and simpler baseline," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3937–3946.

[27] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.

[28] W. Wei, D. Meng, Q. Zhao, Z. Xu, and Y. Wu, "Semi-supervised transfer learning for image rain removal," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3877–3886.

[29] R. Yasarla and V. M. Patel, "Uncertainty guided multi-scale residual learning-using a cycle spinning CNN for single image de-raining," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8405–8414.

[30] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4133–4141.

[31] A. Romero, N. Ballas, S. Ebrahimi Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," 2014, *arXiv:1412.6550*. [Online]. Available: http://arxiv.org/abs/1412.6550

[32] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proc. NIPS*, 2014, pp. 2654–2662.

[33] J. Liu, W. Yang, S. Yang, and Z. Guo, "Erase or fill? Deep joint recurrent rain removal and reconstruction in videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3233–3242.

[34] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[35] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2403–2412.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

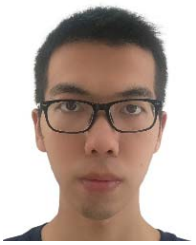
[37] K. Ma *et al.*, "Waterloo exploration database: New challenges for image quality assessment models," *IEEE Trans. Image Process.*, vol. 26, no. 2, pp. 1004–1016, Feb. 2017.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2014, pp. 1–15.

[39] S. Gu, D. Meng, W. Zuo, and L. Zhang, "Joint convolutional analysis and synthesis sparse representation for single image layer separation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1708–1716.

[40] T. Wang, X. Yang, K. Xu, S. Chen, Q. Zhang, and R. W. H. Lau, "Spatial attentive single-image deraining with a high quality real rain dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12270–12279.

[41] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. ECCV*, 2018, pp. 801–818.



Huangxing Lin received the B.S. degree from Beijing Jiaotong University in 2015, and the M.S. degree from Xiamen University, Xiamen, China, in 2018, where he is currently pursuing the Ph.D. degree in signal and information processing. His research interests include image processing, medical image analysis, and machine learning.

Yanlong Li received the B.S. degree from Zhejiang Normal University, China, in 2018. He is currently pursuing the master's degree with the Department of Informatics and Communication Engineering, Xiamen University, China. His research interest includes semantic segmentation.

Xueyang Fu (Member, IEEE) received the Ph.D. degree in signal and information processing from Xiamen University in 2018. He was a Visiting Student at Columbia University, sponsored by the China Scholarship Council. He is currently an Associate Researcher with the Department of Automation, University of Science and Technology of China. His current research interests include machine learning and image processing.

Xinghao Ding (Member, IEEE) was born in Hefei, China, in 1977. He received the B.S. and Ph.D. degrees from the Department of Precision Instruments, Hefei University of Technology, Hefei, in 1998 and 2003, respectively. He was a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA, from 2009 to 2011. Since 2011, he has been a Professor with the School of Information Science and Engineering, Xiamen University, Xiamen, China. His main research interests include machine learning, representation learning, medical image analysis, and computer vision.

Yue Huang (Member, IEEE) received the B.S. degree from Xiamen University, Xiamen, China, in 2005, and the Ph.D. degree from Tsinghua University, Beijing, China, in 2010. She was a Visiting Scholar with Carnegie Mellon University from 2015 to 2016. She is currently an Associate Professor with the Department of Communication Engineering, School of Information Science and Engineering, Xiamen University. Her main research interests include machine learning and image processing.

John Paisley (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Duke University, Durham, NC, USA. He was a Postdoctoral Researcher with Computer Science Department, University of California at Berkeley and Princeton University. He is currently an Associate Professor with the Department of Electrical Engineering, Columbia University, New York, NY, USA, where he is also a member of the Data Science Institute. His current research interest includes machine learning, focusing on models and inference techniques for text and image processing applications.