Clearing the Skies: A Deep Network Architecture for Single-Image Rain Removal

Xueyang Fu, Jiabin Huang, Xinghao Ding, Yinghao Liao, and John Paisley

Abstract—We introduce a deep network architecture called DerainNet for removing rain streaks from an image. Based on the deep convolutional neural network (CNN), we directly learn the mapping relationship between rainy and clean image detail layers from data. Because we do not possess the ground truth corresponding to real-world rainy images, we synthesize images with rain for training. In contrast to other common strategies that increase depth or breadth of the network, we use image processing domain knowledge to modify the objective function and improve deraining with a modestly sized CNN. Specifically, we train our DerainNet on the detail (high-pass) layer rather than in the image domain. Though DerainNet is trained on synthetic data, we find that the learned network translates very effectively to real-world images for testing. Moreover, we augment the CNN framework with image enhancement to improve the visual results. Compared with the state-of-the-art single image de-raining methods, our method has improved rain removal and much faster computation time after network training.

Index Terms—Rain removal, deep learning, convolutional neural networks, image enhancement.

I. INTRODUCTION

THE effects of rain can degrade the visual quality of images and severely affect the performance of outdoor vision systems. Under rainy conditions, rain streaks create not only a blurring effect in images, but also haziness due to light scattering. Effective methods for removing rain streaks are required for a wide range of practical applications, such as image enhancement and object tracking. We present the first deep convolutional neural network (CNN) tailored to this task

X. Fu, J. Huang, X. Ding, and Y. Liao are with the Fujian Key Laboratory of Sensing and Computing for Smart City, School of Information Science and Engineering, Xiamen University, Xiamen 361005, China (e-mail: fxy@stu.xmu.edu.cn; huangjiabin@stu.xmu.edu.cn; yhliao@xmu.edu.cn; dxh@xmu.edu.cn).

J. Paisley is with the Department of Electrical Engineering, Data Science Institute, Columbia University, New York, NY 10027 USA (e-mail: jpaisley@columbia.edu).

Fig. 1. An example real-world rainy image and our de-rained result. (a) Input rainy image. (b) Our result.

and show how the CNN framework can obtain state-of-the-art results. Figure 1 shows an example of a real-world testing image degraded by rain and our de-rained result.

In the last few decades, many methods have been proposed for removing the effects of rain on image quality. These methods can be categorized into two groups: video-based methods and single-image based methods. We briefly review these approaches to rain removal, then discuss the contributions of our proposed DerainNet.

A. Related Work: Video v.s. Single-Image Based Rain Removal

Due to the redundant temporal information that exists in video, rain streaks can be more easily identified and removed in this domain [1]–[4]. For example, Garg and Nayar [1] first propose a rain streak detection algorithm based on a correlation model. After detecting the location of rain streaks, the method uses the average pixel value taken from the neighboring frames to remove streaks. Barnum *et al.* [2] analyze the properties of rain and establish a model of visual effect of rain in frequency space. In [3], the histogram of streak orientation is used to detect rain and a Gaussian mixture model is used to extract the rain layer. In [4], based on the minimization of registration error between frames, phase congruency is used to detect and remove the rain streaks. Many of these methods work well, but are significantly aided by the temporal content of video. In this paper we instead focus on removing rain from a single image.

Compared with video-based methods, removing rain from individual images is much more challenging since much less information is available for detecting and removing rain streaks. Single-image based methods have been proposed to deal with this challenging problem, but success is less noticeable than in video-based algorithms, and there is still

much room for improvement. To give three examples, in [5] rain streak detection and removal is achieved using kernel regression and a non-local mean filtering. In [6], a related work based on deep learning was introduced to remove static raindrops and dirt spots from pictures taken through windows. This method uses a different physical model from the one in this paper. As our later comparisons show, this physical model limits its ability to transfer to rain streak removal. In [7], a generalized low-rank model in which rain streaks are assumed to be low rank is proposed. Both single-image and video rain removal can be achieved by characterizing spatio-temporally correlations of rain streaks.

Recently, several methods based on dictionary learning have been proposed [8]–[12]. In [9], the input rainy image is first decomposed into its base layer and detail layer. Rain streaks and object details are isolated in the detail layer while the structure remains in the base layer. Then sparse coding dictionary learning is used to detect and remove rain streaks from the detail layer. The output is obtained by combining the de-rained detail layer and base layer. A similar decomposition strategy is also adopted in method [12]. In this method, both rain streaks removal and non-rain component restoration is achieved by using a hybrid feature set. In [10], a self-learning based image decomposition method is introduced to automatically distinguish rain streaks from the detail layer. Luo *et al.* [11] use discriminative sparse coding to recover a clean image from a rainy image. A drawback of methods [9], [10] is that they tend to generate over-smoothed results when dealing with images containing complex structures that are similar to rain streaks, as shown in Figure 9(c), while method [11] usually leaves rain streaks in the de-rained result, as shown in Figure 9(d). Moreover, all four dictionary learning based frameworks [9]–[12] require significant computation time. More recently, patch-based priors for both the clean and rain layers have been explored to remove rain streaks [13]. In this method, the multiple orientations and scales of rain streaks are addressed by pre-trained Gaussian mixture models.

B. Contributions of our DerainNet Approach

As mentioned, compared to video-based methods, removing rain from a single image is significantly more difficult. This is because most existing methods [9]–[11], [13] only separate rain streaks from object details by using low level features, for example by learning a dictionary for object representation. When an object's structure and orientation are similar with that of rain streaks, these methods have difficulty simultaneously removing rain streaks and preserving structural information. Humans on the other hand can easily distinguish rain streaks within a single image using high-level features such as context information. We are therefore motivated to design a rain detection and removal algorithm based on the deep convolutional neural network (CNN) [14], [15]. CNN's have achieved success on several low level vision tasks, such as image denoising [16], super-resolution [17], [18], image deconvolution [19], image inpainting [20] and image filtering [21]. We show that the CNN can also provide excellent performance for single-image rain removal.

In this paper, we propose "DerainNet" for removing rain from single-images, which we base on the deep CNN. To our knowledge, this is the first approach based on deep learning to directly address this problem. Our main contributions are threefold:

1) DerainNet learns the nonlinear mapping function between clean and rainy detail (i.e., high resolution) layers directly and automatically from data. Both rain removal and image enhancement are performed to improve the visual effect. We show significant improvement over three recent state-of-the-art methods. Additionally, our method has significantly faster testing speed than the competitive approaches, making it more suitable for real-time applications.

2) Instead of using common strategies such as increasing neurons or stacking hidden layers to effectively and efficiently approximate the desired mapping function, we use image processing domain knowledge to modify the objective function and improve the de-rain quality. We show how better results can be obtained without introducing more complex network architecture or more computing resources.

3) Because we lack access to the ground truth for real-world rainy images, we synthesize a dataset of rainy images using real-world clean images, which we can take as the ground truth. We show that, though we train on synthesized rainy images, the resulting network is very effective when testing on real-world rainy images. In this way, the model can be learned with easy access to an unlimited amount of training data.

II. DERAINNET: DEEP LEARNING FOR RAIN REMOVAL

We illustrate the proposed DerainNet framework in Figure 2. As discussed in more detail below, we decompose each image into a low-frequency base layer and a high-frequency detail layer. The detail layer is the input to the CNN for rain removal. To further improve visual quality, we introduce an image enhancement step to sharpen the results of both layers since the effects of heavy rain naturally leads to a hazy effect.

A. Training on High-Pass Detail Layers

We denote the input rainy image and corresponding clean image as $\mathbf{I}$ and $\mathbf{J}$ respectively. Initially, a goal may be to train a network architecture $h_{\mathbf{P}}(\cdot)$ that minimizes

$$L = \frac{1}{N} \sum_{n=1}^{N} \| f_{\mathbf{W}}(\mathbf{I}^n) - \mathbf{J}^n \|_F^2, \qquad (1)$$

where $\mathbf{W}$ are the network parameters and F is the Frobenius norm and n indexes the image. However, we found that the result obtained by directly training in the image domain is not satisfactory. In Figure 3(a), we show an example of a synthetic rainy image. Note that this image is used in the training process. In Figure 3(b) we see that even when this image is used as a training sample, the de-rained image still exhibits clear rain streaks when zoomed in.

Figure 3(b) implies that the desired mapping function was not well learned when training on the image domain, i.e., the
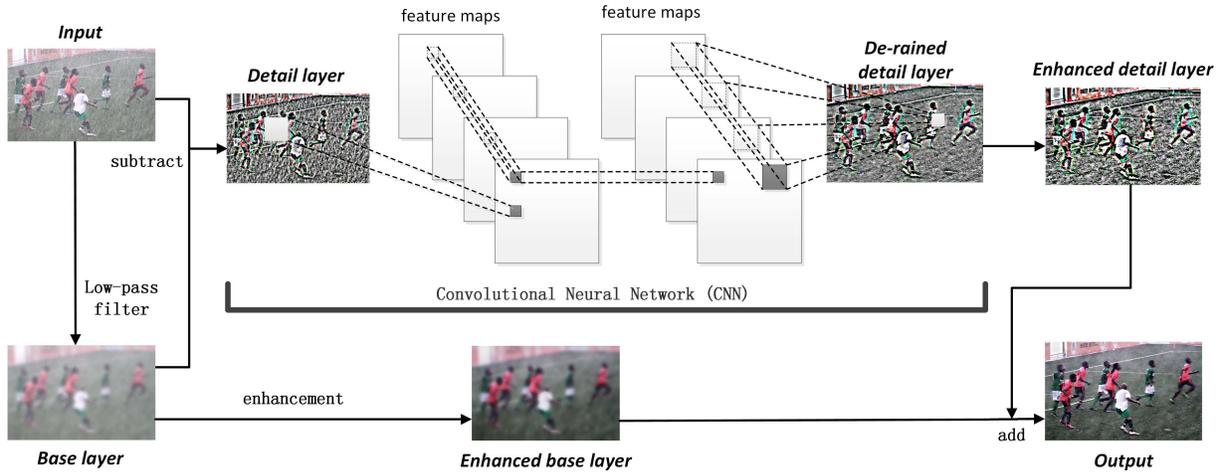
Fig. 2. The proposed DerainNet framework for single-image rain removal. The intensities of the detail layer images have been amplified for better visualization.
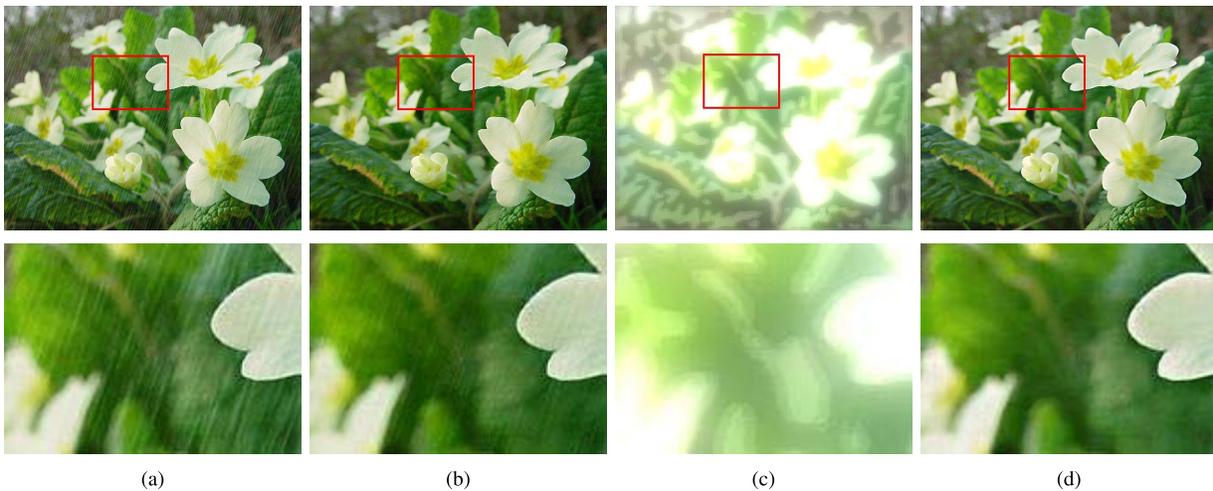

(a) (b) (c) (d)

Fig. 3. CNN learning options: (b) directly on image domain with depth = 3 (equivalent to retraining [6] on new data), (c) directly on image domain with depth = 10, and (d) on high-frequency detail layer with depth = 3. The first row shows the full image and the second row a zoomed-in region. (a) Rainy image. (b) Image domain, depth = 3. (c) Image domain, depth = 10. (d) Detail layer domain, depth = 3.

model under-fit the data. It is natural to ask whether it is necessary to train a more complex model to further improve the capacity of the network. As is well-known, there are two ways to improve a network's capacity in the deep learning domain. One way is to increase the depth of network [22] by stacking more hidden layers. Usually, more hidden layers can help to obtain high-level features. However, the de-rain problem is a low-level image task and the deeper structure is not necessarily better for this image processing problems. Furthermore, training a feed-forward network with more layers suffers from gradient vanishing unless other training strategies or more complex network structures are introduced. As shown in Figure 3(c), when we add network depth to improve the modeling ability, the result actually becomes worse. The other approach is to increase the breadth of network [23] by using more neurons in each hidden layer. However, to avoid over-fitting, this strategy requires more training data and computation time that may be intolerable under normal computing condition.

To address these issues for the de-raining problem, we instead use a priori image processing knowledge to modify the objective function rather than increase the complexity of the problem. Conventional end-to-end procedures directly uses image patches to train the model by finding a mapping function f that transforms the input to output [6], [17]. Motivated by Figure 3, rather than directly train on the image, we first decompose the image into the sum of a "base" layer and a "detail" layer by using a low-pass filter,

$$\mathbf{J} = \mathbf{J}_{\text{base}} + \mathbf{J}_{\text{detail}}. \qquad (2)$$

Using on image processing techniques, we found that after applying an appropriate low-pass filters such as [24], [25], and [26], low-pass versions of both the rainy image $\mathbf{I}_{\text{base}}$ and the clean image $\mathbf{J}_{\text{base}}$ are smooth and are approximately equal, as shown in Figure 4. In other words, both the rain streaks and the object's details remain in the high-pass detail layer and $\mathbf{I}_{\text{base}} \approx \mathbf{J}_{\text{base}}$. This implies that the base layer portion can be removed from the training process, significantly

Fig. 4. Example base and detail layers of two synthesized images. We use the guided filtering [24] as the low-pass filter to generate the results. (a) Clean images $\mathbf{J}$. (b) Rainy images $\mathbf{I}$. (c) $\mathbf{J}_{\text{base}}$. (d) $\mathbf{I}_{\text{base}}$. (e) $|\mathbf{J}_{\text{base}} - \mathbf{I}_{\text{base}}|$.
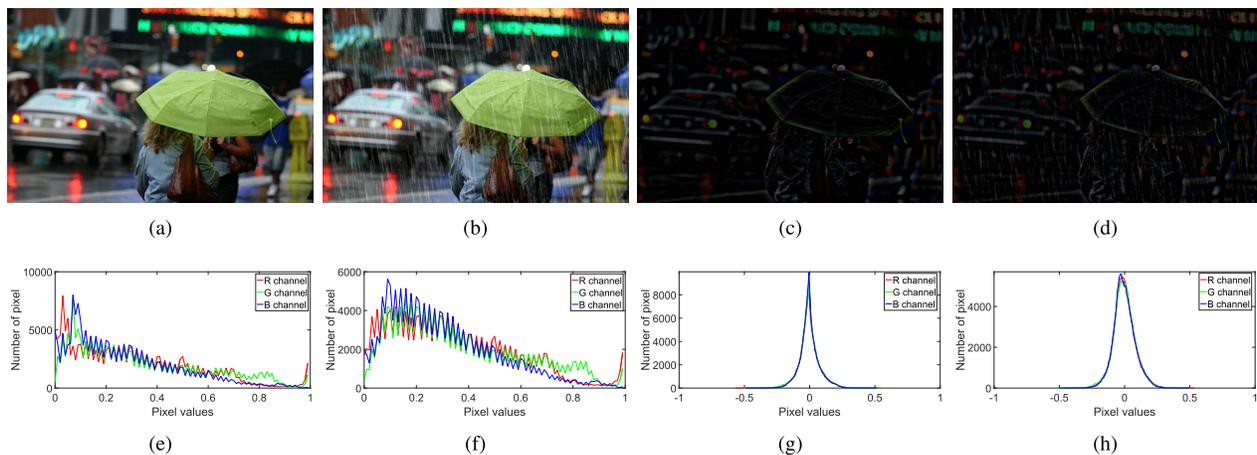

Fig. 5. Sparsity of detail layer. The detail layers are obtained by $\mathbf{J}_{\text{detail}} = \mathbf{J} - \mathbf{J}_{\text{base}}$ and $\mathbf{I}_{\text{detail}} = \mathbf{I} - \mathbf{I}_{\text{base}}$. (a) Clean image $\mathbf{J}$. (b) Rainy image $\mathbf{I}$. (c) $\mathbf{J}_{\text{detail}}$. (d) $\mathbf{I}_{\text{detail}}$. (e) Histogram of (a). (f) Histogram of (b). (g) Histogram of (c). (h) Histogram of (d).

simplifying the mapping needed to be learned by the CNN. Thus, we rewrite the objective function in (1) as

$$L = \frac{1}{N} \sum_{n=1}^{N} \left\| f_{\mathbf{W}}(\mathbf{I}_{\text{detail}}^n) - \mathbf{J}_{\text{detail}}^n \right\|_F^2 . \tag{3}$$

This directly lead us to train the CNN network on the detail layer instead of the image domain. Moreover, training on the detail layer has several advantages. First, after subtracting the base layer, the detail layer is sparser than the image since most regions in the detail layer are close to zero. As shown in Figure 5, the detail layer has many more pixels that are close to zero than the image itself. Taking advantage of the sparsity of the detail layer is a widely used technique in existing de-raining methods [9]–[11]. In the context of a neural network, training a CNN on the detail layer also follows the procedure of mapping an input patch to an output patch, but since the mapping range has been significantly decreased, the regression problem is significantly easier to handle for a deep learning model. Thus, training on the detail layer instead of the image domain can improve learning the network weights and thus the de-raining result without a large increase in training data or computational resources.

A second advantage of training on sparse data is that it can improve the convergence of the CNN. As we show in our experiments (Figure 17), training on the detail layer converges much faster than training on the image domain. A third advantage is that decomposing an image into base and detail layers is widely used by the wider image enhancement community [27], [28]. These enhancement procedures are tailored to this decomposition and can be easily embedded into our architecture to further improve image quality, which we describe in Section II-D.

We therefore first decompose the image into a base layer by using a low-pass filter and a detail layer; the detail layer is equal to the difference between the image and the base layer. We use the guided filtering method of [24] as the low-pass filter because it is simple and fast to implement. In this paper, the guidance image is the input image itself. However, the choice of low-pass filter is not limited to guided filtering; other filtering approaches were also effective in our experiments, such as bilateral filtering [25] and rolling guidance filtering [26]. Results with these filters were nearly identical, so we choose [24] for its low computational complexity.

After this decomposition we train the CNN on the detail layer image instead of raw image itself according
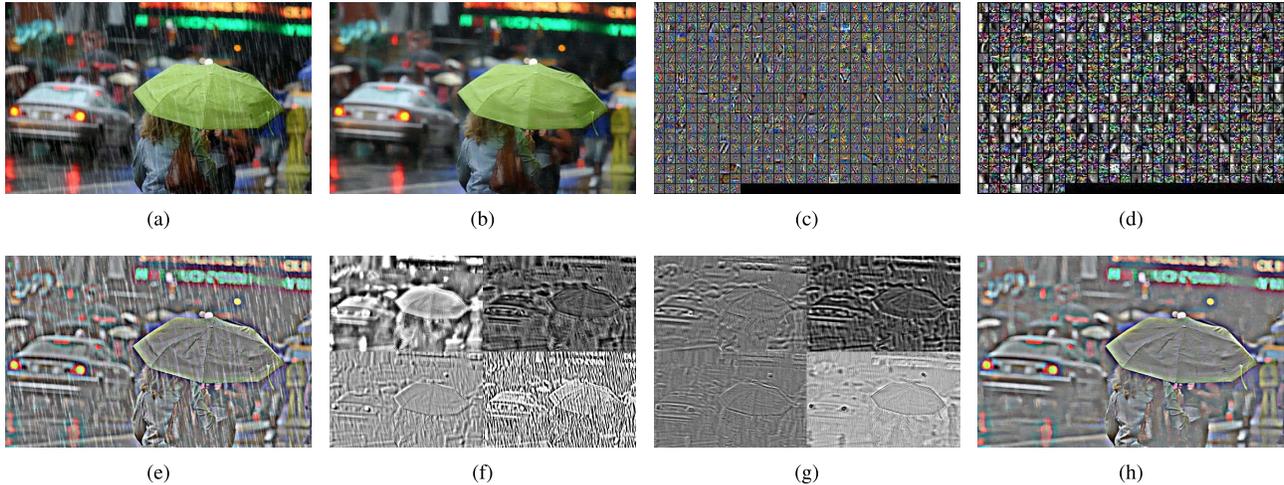
Fig. 6.   Visualization of intermediate results. The first row shows our de-raining result and the trained weights $\mathbf{W}^1$ (512 kernels of size $16 \times 16 \times 3$) and $\mathbf{W}^3$ (3 kernels of size $8 \times 8 \times 512$, one for each color channel). For $\mathbf{W}^3$ we visualize these three kernels as RGB images across the 512 dimensions. Since the 512 kernels $\mathbf{W}^2$ are $1 \times 1 \times 512$, we do not show them. The second row shows the corresponding hidden layer activations. (e) and (h) show the detail layer input and output of the network. In (f) we show four of the 512 convolutional output of (e) in the first layer. These appear to be producing different "views" of the rain. In (g) we show four of the 512 layers that are combined to produced the three layer RGB output in (f). The intensities of the images in the second row have been amplified for better visualization.

Fig. 7.   An example of synthesized rainy images. The top left is the clean image and the remaining are various images synthesized.

to Eq. (3). This step represents the CNN portion of Figure 2. In Figure 3(d) we show an example of the de-rained image using this training approach. In terms of rain streak removal, the result is clearly better than the same CNN structure trained on the image domain shown in Figure 3(b). This conclusion is further supported by our experiments below.

B. Our Convolutional Neural Network

We build our network structure same to [6]. Our network structure can be expressed as three operations:

$$f^l(\mathbf{I}_{\text{detail}}) = \sigma\left(\mathbf{W}^l * f^{l-1}(\mathbf{I}_{\text{detail}}) + \mathbf{b}^l\right), \quad l = 1, 2 \quad (4)$$

$$f_{\mathbf{W}}(\mathbf{I}_{\text{detail}}) = \mathbf{W}^l * f^{l-1}(\mathbf{I}_{\text{detail}}) + \mathbf{b}^l, \quad l = 3, \quad (5)$$

where l indexes layer number, $*$ indicates the convolution operation and b^l is the bias. We define $\sigma(\cdot)$ to be the nonlinear hyperbolic tangent function and $f^0(\mathbf{I}_{\text{detail}}) = \mathbf{I}_{\text{detail}}$. We use two hidden layers in our DerainNet architecture and Eq. (5) is the output of the cleaned detail layer.

To better understand the effects of the network $f_{\mathbf{W}}$, we show the learned weights and intermediate results from the hidden layers in Figure 6. The first hidden layer performs feature extraction on the input detail layer, which is similar to the common strategy used for image restoration of extracting and representing image patches by a set of dictionary elements. Thus, $\mathbf{W}^1$ contains some filters that look like edge detectors that align with the direction of rain streaks and object edges. The second hidden layer performs the rain streaks removal and $f^2(\mathbf{I}_{\text{detail}})$ looks smoother than $f^1(\mathbf{I}_{\text{detail}})$. The third layer performs reconstruction and enhances the smoothed details with respect to image content. As can be seen in Figure 6(h), $f_{\mathbf{W}}(\mathbf{I}_{\text{detail}})$ contains clear details with most of the rain removed. The intermediate results show that the CNN is effective at feature extraction and helps to recognize and remove rain streaks.

C. Training

We use stochastic gradient descent (SGD) to minimize the objective function in Eq. (3). Since it is extremely difficult to obtain numerous clean/rainy image pairs from real-world data, we synthesize rain using Photoshop[1] to create our training dataset. We randomly collected a total of 350 clean outdoor images from the UCID dataset [29], the BSD dataset [30] and Google image search which we used to synthesize rainy images. Each clean image was used to generate 14 rainy images of different streak orientations and intensity. An example is shown in Figure 7. Thus, we create a dataset containing

[1]http://www.photoshopessentials.com/photo-effects/rain/

$350 \times 14 = 4900$ rainy images, each having a corresponding ground truth clean image. We randomly selected one million 64×64 clean/rainy patch pairs from this synthesized data as training samples. A 56×56 output is generated to avoid border effects caused by convolution. In each iteration, t, the CNN weight and bias are updated using back-propagation,

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \alpha(f_{\mathbf{W}}(\mathbf{I}_{\text{detail\_i}}) - \mathbf{J}_{\text{detail\_i}})^T \frac{\partial f_{\mathbf{W}}(\mathbf{I}_{\text{detail\_i}})}{\partial \mathbf{W}},$$
$$\mathbf{b}_{t+1} = \mathbf{b}_t - \alpha(f_{\mathbf{W}}(\mathbf{I}_{\text{detail\_i}}) - \mathbf{J}_{\text{detail\_i}})^T, \quad (6)$$

where α is the learning rate and $(\mathbf{I}_{\text{detail\_i}}, \mathbf{J}_{\text{detail\_i}})$ is the ith patch pair.

D. Combining CNN With Image Enhancement

After training the network, the de-rained image can be obtained by directly adding the output detail layer to the base layer,

$$\mathbf{O} = \mathbf{I}_{\text{base}} + f_{\mathbf{W}}(\mathbf{I}_{\text{detail}}), \quad (7)$$

where $\mathbf{O}$ is the de-rained output. However, when dealing with heavy rain the result unsurprisingly looks hazy, as shown in Figure 8(b). Fortunately, we can easily embed image enhancement technology into our framework to create a better visual result. Different mature and advanced image enhancement algorithms can be directly adopted in this framework as post-processing. In this paper, we use the non-linear function [31] to enhance the base layer, and boost the detail layer by simply multiplying the output of the CNN by two to magnify the details,

$$\mathbf{O}_{\text{enhanced}} = (\mathbf{I}_{\text{base}})_{\text{enhanced}} + 2 f_{\mathbf{W}}(\mathbf{I}_{\text{detail}}), \quad (8)$$

where $\mathbf{O}_{\text{enhanced}}$ is the de-rained output with enhancement and $(\mathbf{I}_{\text{base}})_{\text{enhanced}}$ is the enhanced base layer. Figure 8(c) shows the de-rained result with image enhancement. As shown in the intermediate results in Figures 8(d)-(g), virtually all of rain removal is being performed on the detail layer by the CNN, while the image enhancement on the base layer improves the global contrast and leads to a better visual result than shown in Figure 8(b) without using enhancement.

III. EXPERIMENTS

To evaluate our DerainNet framework, we test on both synthetic and real-world rainy images. As mentioned previously, both testing frameworks are performed using the network trained on synthesized rainy images. We compare with three recent high quality de-raining methods [10], [11], [13]. Software implementations of these methods were provided in Matlab by the authors. We use the default parameters reported in these three papers. All experiments are performed on a PC with Intel Core i5 CPU 4460, 8GB RAM and NVIDIA Geforce GTX 750. Our network contains two hidden layers and one output layer as described in Section II-B. We set kernel sizes $s_1 = 16, s_2 = 1$ and $s_3 = 8$, respectively. The number of feature maps for each hidden layer are $n_1 = n_2 = 512$. We set the learning rate to $\alpha = 0.01$. Our rainy image dataset and Matlab implementation can be found at http://smartdsp.xmu.edu.cn/derainNet.html.
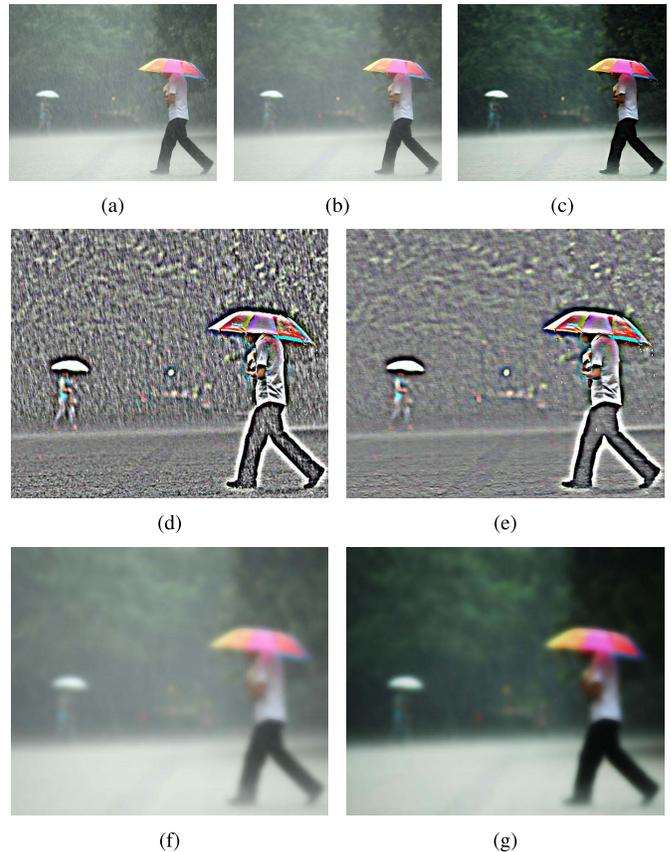

Fig. 8. Visualization of intermediate results with enhancement. Intensities of detail layers have been amplified for better visualization. (a) Rainy image. (b) Result by Eq. (7). (c) Result by Eq. (8). (d) Detail layer. (e) De-rained (d). (f) Base layer. (g) Enhanced (f).
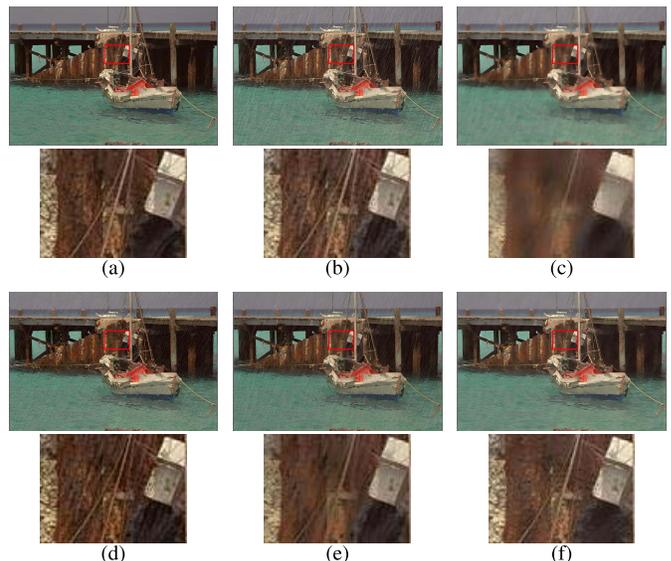
Fig. 9. Results on synthesized rainy image "dock". Row 2 shows corresponding enlarged parts of red boxes in Row 1. (a) Ground truth. (b) Rainy image. (c) Method [10]. (d) Method [11]. (e) Method [13]. (f) Our result.

A. Synthesized Data

We first evaluate the results of testing on newly synthesized rainy images. In our first results, we synthesize new rainy images by selecting from the set of 350 clean images from our database. Figure 9 shows visual comparisons for one such

| (a) | (b) | (c) | (d) | (e) | (f) |

Fig. 10.   Example results on synthesized rainy images "umbrella", "rabbit", "girl" and "bird." These rainy images were for testing and not used for training. (a) Ground truth. (b) Synthesized image. (c) Method [10]. (d) Method [11]. (e) Method [13]. (f) DerainNet.

TABLE I

QUANTITATIVE MEASUREMENT RESULTS USING SSIM ON SYNTHESIZED TEST IMAGES

Images	Ground truth	Rainy image	Method [10]	Method [11]	Method [13]	Ours
dock	1	0.86	0.84	0.88	0.90	**0.92**
umbrella	1	0.75	0.83	0.81	0.86	**0.88**
rabbit	1	0.72	0.74	0.77	0.79	**0.85**
girl	1	0.93	0.80	**0.94**	0.91	**0.94**
bird	1	0.57	0.63	0.62	0.75	**0.82**
100 new images	1	0.79 ± 0.13	0.73 ± 0.07	0.84 ± 0.09	0.82 ± 0.10	**0.89 ± 0.06**
Rain12 [13]	1	0.84 ± 0.07	0.69 ± 0.07	0.80 ± 0.07	0.89 ± 0.04	**0.90 ± 0.04**

synthesized test image. As can be seen, method [10] exhibits over-smoothing of the rope and method [11], [13] leaves significant rain streaks in the result. This is because [10], [11], [13] are algorithms based on low-level image features. When the rope's orientation and magnitude is similar with that of rain, methods [10], [11], [13] cannot efficiently distinguish the rope from rain streaks. However, as shown in the last result, the multiple convolutional layers of DerainNet can identify and remove rain while preserving the rope.

Figure 10 shows visual comparisons for four more synthesized rainy images using different rain streak orientations and magnitudes. Since the ground truth is known, we use the structure similarity index (SSIM) [32] for quantitative evaluation. A higher SSIM value indicates a de-rained image that is closer to the ground truth in terms of image structural properties. (For the ground truth, the SSIM equals 1.) For a fair comparison, the image enhancement operation is *not* implemented by our algorithm for these synthetic experiments.

As is again evident in these results, method [10] over-smooths the results and methods [11], [13] leave rain streaks,

TABLE II

TIMES (ROW) BEAT (COL)

	[10]	[11]	[13]	Ours
[10]	–	5	4	0
[11]	107	–	74	6
[13]	108	38	–	6
Ours	112	106	106	–

both of which are addressed by our algorithm. Moreover, we see in Table I that our method has the highest SSIM values, in agreement with the visual effect. Also shown in Table I is the performance of the three methods on 100 newly-synthesized testing images using our synthesizing strategy.

In Table I we also show results applying the same trained algorithms for each method on 12 newly synthesized rainy images (called *Rain12*) [13] that are generated using photorealistic rendering techniques [33]. This clearly highlights the generalizability of DerainNet to new scenes; whereas the other algorithms either decrease the performance or leave it unchanged, DerainNet still shows improvement. In Table II we show the number of images for which the algorithm

TABLE III
QUANTITATIVE MEASUREMENT RESULTS OF BIQI ON REAL-WORLD TEST IMAGES

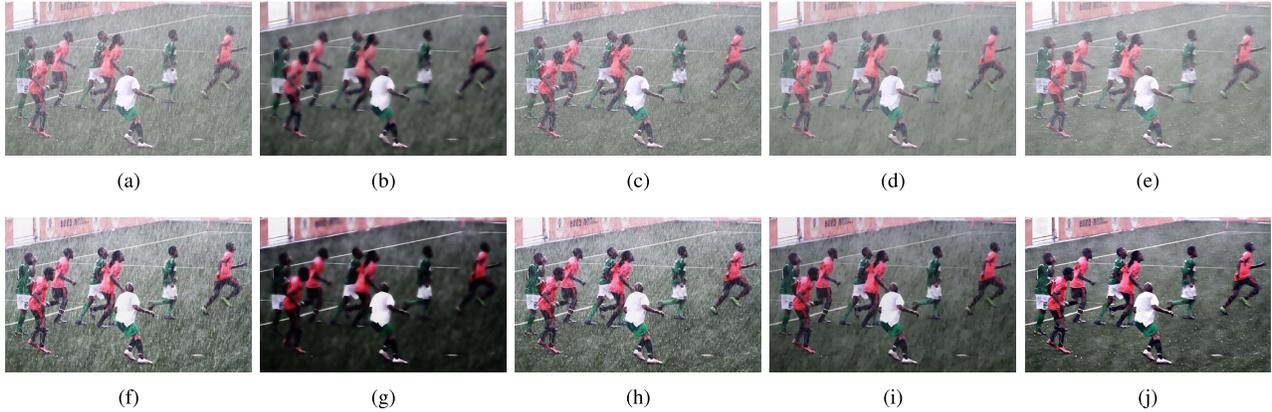| Images	Input	Method [10]	Method [11]	Method [13]	Ours
soccer	57.96	42.70	53.86	35.64	**33.35**
Buddha	49.06	39.55	50.13	39.90	**28.10**
street	37.70	40.67	38.32	38.98	**34.08**
cars	27.84	40.08	**21.17**	31.70	24.18
100 test images	33.00 ± 13.19	39.63 ± 7.66	31.43 ± 9.81	34.60 ± 7.78	**29.86 ± 6.98**

Fig. 11. Comparison of algorithms on a real-world "soccer" image with and without enhancement. (a) Rainy image. (b) Method [10]. (c) Method [11]. (d) Method [13]. (e) Our result. (f) Rainy enhanced. (g) Method [10] enhanced. (h) Method [11] enhanced. (i) Method [13] enhanced. (i) Our result enhanced.
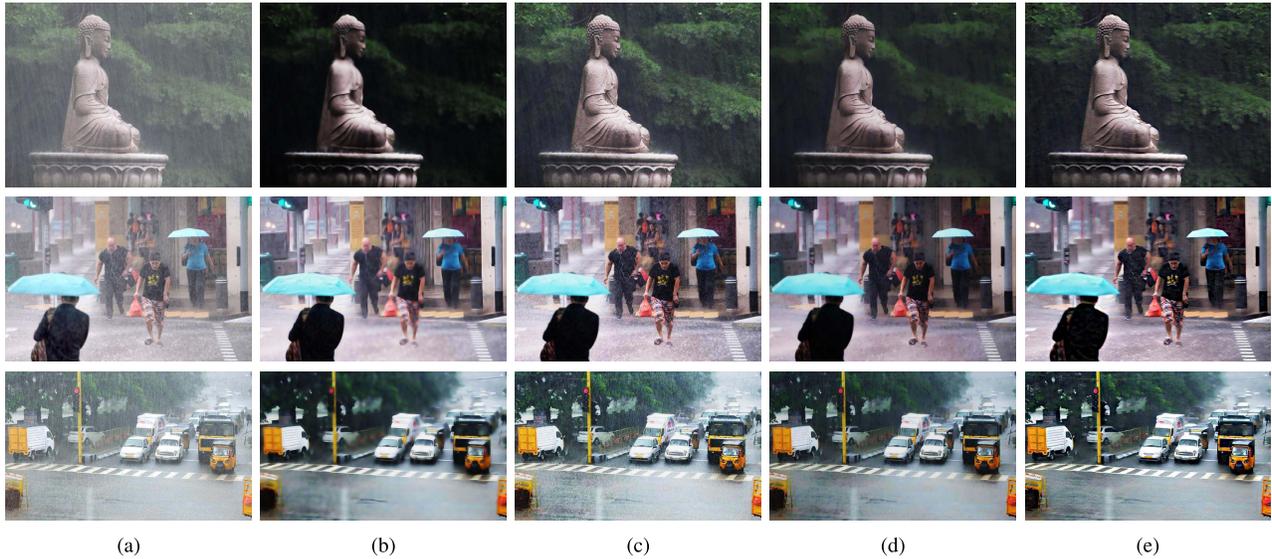

Fig. 12. Three more results on real-world rainy images: (top-to-bottom) "Buddha," "street," "cars." All algorithms use image enhancement. (a) Rainy images. (b) Method [10]. (c) Method [11]. (d) Method [13]. (e) Our results.

on the row outperformed the algorithm on the column for these 112 images.

B. Real-World Data

Since we do not possess the ground truth corresponding to real-world rainy images, we test DerainNet on real-world data using the network trained on the 4900 synthesized images from the previous section. In Figure 11 we show the results of all algorithms with and without enhancement, where enhancement of [10], [11] and [13] are performed as post-processing, and

for DerainNet is performed as shown in Figure 2. In our quantitative comparison below, we use enhancement for all results, but note that the relative performance between algorithms was similar without using enhancement. We show results on three more real-world rainy images in Figure 12. Although we use synthetic data to train our DerainNet, we see that this is sufficient for learning a network that is effective when applied to real-world images.

In Figure 12, the proposed method arguably shows the best visual performance on simultaneously removing rain and preserving details. Since the ground truth is unavailable in

TABLE IV

AVERAGE SCORES OF USER STUDY

Images	Input	Method [10]	Method [11]	Method [13]	Ours
Scores	1.51	1.46	1.73	2.57	**4.11**

TABLE V

AVERAGE SSIM OF DIFFERENT KERNEL SIZES

Kernel sizes	4-1-2	8-1-4	16-1-8 (default)
SSIM	0.84 ± 0.06	0.87 ± 0.05	0.89 ± 0.06

Kernel sizes	16-3-8	16-5-8
SSIM	0.89 ± 0.06	0.90 ± 0.07

TABLE VI

AVERAGE SSIM OF DIFFERENT NETWORK WIDTH

Width	64	128	256	512 (default)
SSIM	0.80 ± 0.05	0.82 ± 0.05	0.85 ± 0.05	**0.89 ± 0.06**

TABLE VII

AVERAGE SSIM OF DIFFERENT NETWORK DEPTH

Depth	3 (default)	5	10
SSIM	**0.89 ± 0.06**	0.84 ± 0.05	0.79 ± 0.04

these examples, we cannot definitively say which algorithm performs quantitatively the best. Instead, we use a reference-free measure called the Blind Image Quality Index (BIQI) [34] for quantitative evaluation. This index is designed to provide a score of the quality of an image without reference to ground truth. A lower value of BIQI indicates a higher quality image. However, as with all reference-free image quality metrics, BIQI is arguably not always subjectively correct. Still, as Table III indicates, our method has the lowest BIQI on 100 newly obtained real-world testing images. This gives additional evidence that our method outputs an image with greater improvement.

To provide realistic feedback and quantify the subjective evaluation of DerainNet, we also constructed an independent user study. In this experiment, we use the de-rained results (with enhancement) of the same 100 real-world images scored with BIQI. For each image, we randomly order the outputs of the four algorithms, as well as the original rainy image, and display them on a screen. We then separately asked 20 participants to rank each image from 1 to 5 subjectively according to quality, with the instructions being that visible rain should decrease the quality and clarity should increase quality (1 represents the worst quality image and 5 represents the best quality image). We show the average scores in Table IV from these 2000 trials. As is evident, methods [10] and [11] do not make a clear improvement over the original image. Method [7] does clearly improve the rainy image, but the proposed method is subjectively superior to all images. This small-scale experiment gives additional support along with BIQI and our own subjective assessment that DerainNet improves the de-raining on real-world images.

C. Parameter Settings

In this section, we test different parameters setting to study their impact on performance. We use the same training data as previously. The testing data includes the same 100 newly-synthesized images as well as the new *Rain12* images [13].

1) Kernel Size: First, we test the impact of different kernel sizes. The default kernel sizes for the three levels are 16, 1 and 8; we denote this network as 16-1-8. We fix the kernel size of the second layer and reduce the kernel sizes of first and third layers to 4-1-2 and 8-1-4. We then performed experiments by instead increasing the kernel size of second layer to 16-3-8 and 16-5-8. Table V shows the average SSIM values for these different kernel sizes. As can be seen, larger kernel sizes can generate better results. This is because more structure and texture can be modeled using a large kernel. On the contrary, from our experiments we find that increasing the kernel size of the second layer brings only limited improvement. This is because the second layer performs a non-linear operation for rain removal and the 1×1 kernel can achieve promising

results. Thus, we choose 16-1-8 as the default setting of kernel size.

2) Network Width: Intuitively, if we increase the network width by increasing the number of kernels, n_1 and n_2, the performance should improve. We train three models by using the values: $n_1, n_2 \in \{64, 128, 256\}$ and compare them to our default setting of $n_1 = n_2 = 512$. Table VI shows the average SSIM values for these four models. As can be seen, better performance can be achieved by increasing the width of the network. However, increasing the number of kernels improves the performance at the cost of running time since more convolutional operations are required. Thus, we choose $n_1 = n_2 = 512$ as the default setting of network width.

3) Network Depth: We also test the performance of using deeper structures by adding more non-linear layers. We train and test on 3 networks with depths 3, 5 and 10. As shown in Table VII, for the de-raining problem, increasing the network depth does not bring better results using a feed-forward network structure. This is a result of gradient vanishing, which may be addressed by designing a more complex network structure (with increased computation time). However, our DerainNet generates high quality results with only 3 layers as a result of our proposed detail training strategy, and so the complexity and computation time of the model can be significantly reduced. Therefore, we adopt three layers as the default setting.

D. Comparison With Another Potential Deep Learning Method

The proposed DerainNet combines image domain knowledge as pre-processing before the CNN step. As mentioned [6] proposed directly using a CNN to removing dirt and drops from a window [6]. (This is the only other related deep learning approach we are aware of.) As motivated in Section II-A and Figure 3, directly training on the image domain has drawbacks. We show a few other examples on real and synthesized data in Figure 13. As is evident from these examples as well, directly training on the image domain has drawbacks that

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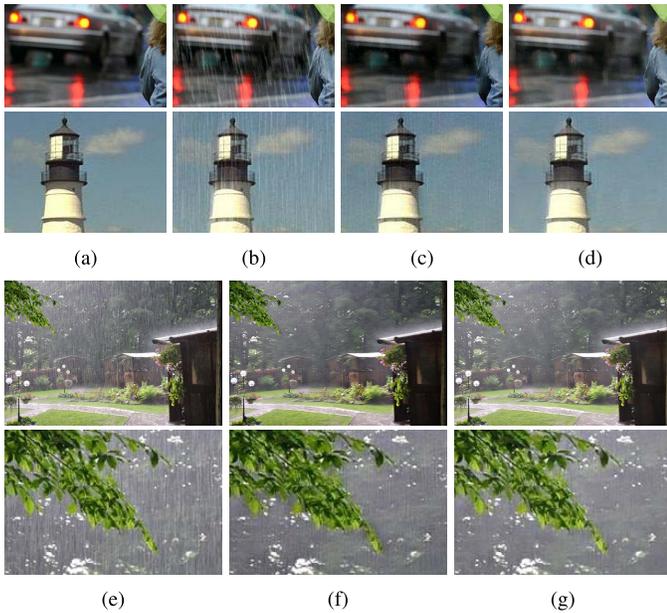

(e)   (f)   (g)

Fig. 13.　Top: Two zoomed-in regions of images synthesized with rain. Bottom: Real world rainy image (no enhancement). (a) Truth. (b) Rain. (c) Plain CNN [6]. (d) Ours. (e) Rainy image. (f) Plain CNN [6]. (g) Ours.
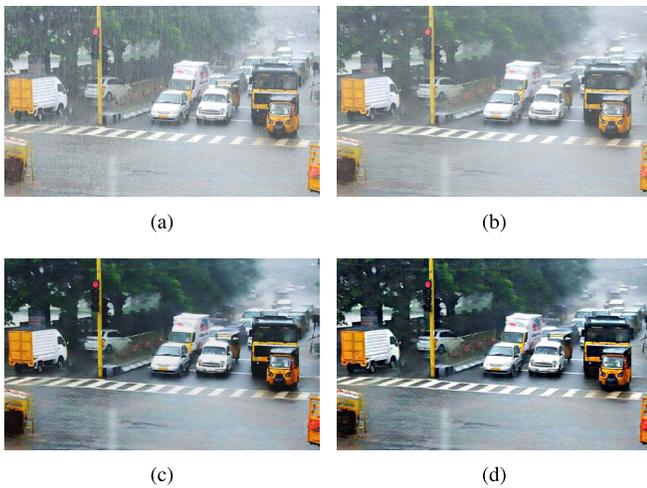

(a)   (b)

(c)   (d)

Fig. 14.　Impact of image enhancement of different processing strategies. (a) Rainy image. (b) De-rain result. (c) Post-processed (b). (d) Simultaneous processing.

are effectively addressed by our approach. We note that both approaches have virtually identical computational complexity.

E. Impact of Image Enhancement Step

In this section we assess the impact of image enhancement on our algorithm. We adopt three processing strategies for real-world data. Specifically, we conduct de-raining without any enhancement, de-raining with the enhancement as a post-processing step after reconstruction, and simultaneous de-raining and enhancement as proposed in Figure 2. Figure 14 shows one example of these different processing strategies. As can be seen, rain streaks are removed by the CNN alone, while the enhancement step further improves the visual quality.

TABLE VIII
BIQI RESULTS FOR THREE ENHANCEMENT STRATEGIES

Images	No enhance	Post-enhance	Proposed
soccer	34.69	**32.06**	33.35
Buddha	37.88	29.72	**28.10**
street	37.47	34.33	**34.08**
cars	31.56	26.08	**24.18**
100 test images	31.86 ± 7.89	29.98 ± 7.82	**29.86 ± 6.98**

(a)   (b)

(c)   (d)   (e)

(f)   (g)   (h)
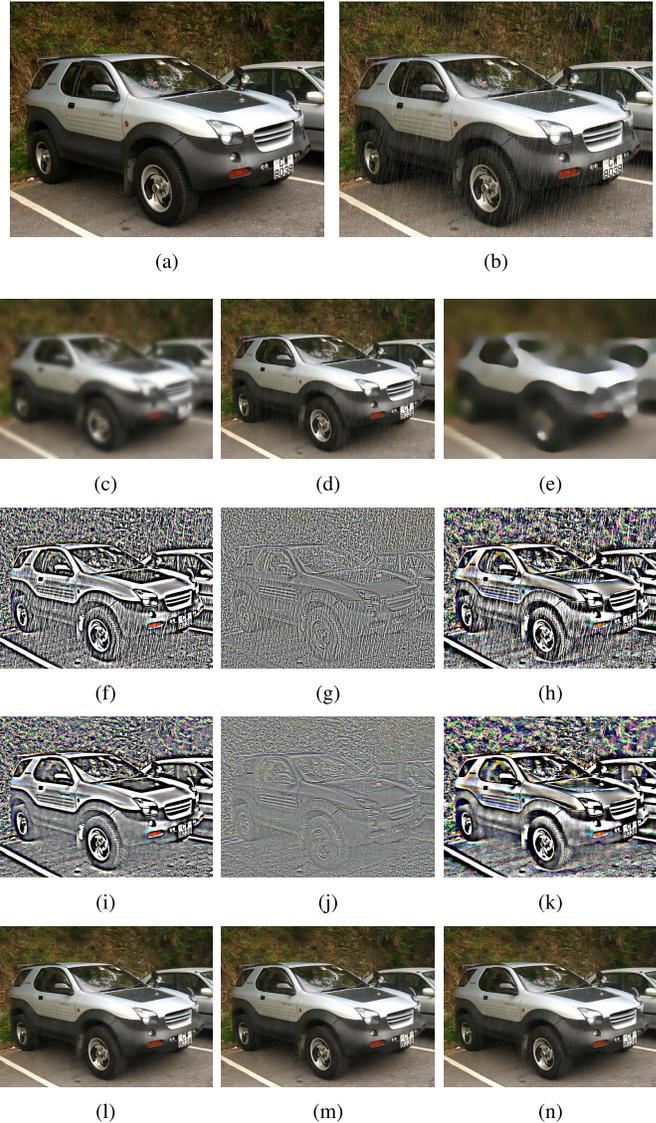

(i)   (j)   (k)

(l)   (m)   (n)

Fig. 15.　Impact of three low-pass filters [24]–[26]. SSIM values of (l), (m) and (n) are 0.9181, 0.9160 and 0.9158, respectively. Intensities of detail layer images have been amplified for better visualization. (a) Clean image. (b) Rainy image. (c) Base layer [24]. (d) Base layer [25]. (e) Base layer [26]. (f) Detail layer [24]. (g) Detail layer [25]. (h) Detail layer [26]. (i) De-rained (f). (j) De-rained (g). (k) De-rained (h). (l) Result of (c)+(i). (m) Result of (d)+(j). (n) Result of (e)+(k).

We also use the BIQI metric to evaluate the three strategies by testing on collected real-world images, as shown in Table VIII. Although the visual quality is similar with post-processing, the overall BIQI shows the best quantitative performance of our "mid-processing" approach.
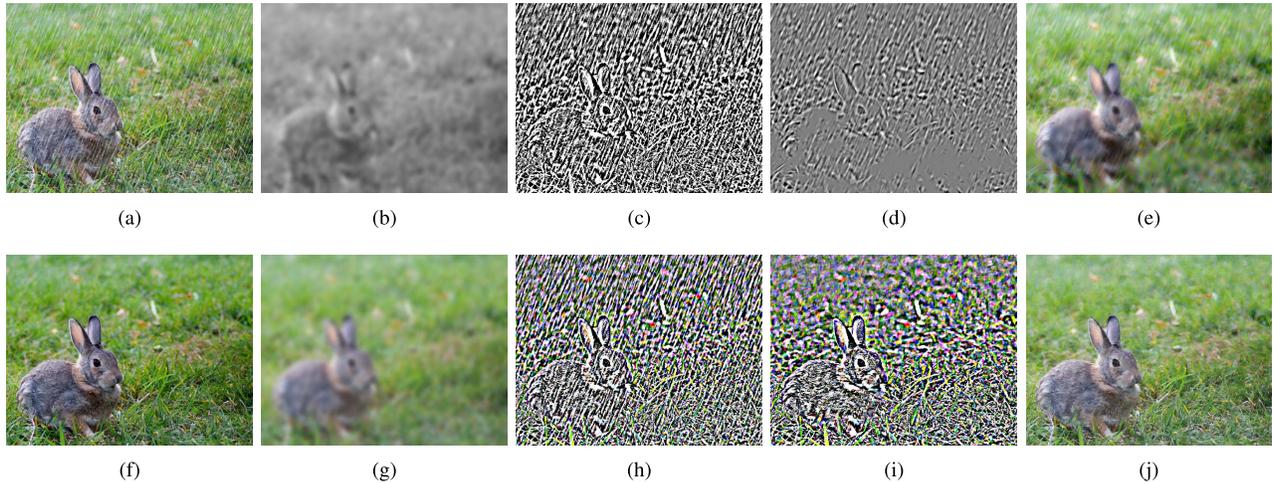
Fig. 16. Comparison with method [10] by using the bilateral filtering approach as pre-processing for the respective models. SSIM values of (e) and (j) are 0.77 and 0.87, respectively. Intensities of detail layer images have been amplified for better visualization. (a) Rainy image. (b) Base layer (Y channel). (c) Detail layer (Y channel). (d) De-rained (c) [10]. (e) De-rained result [10]. (f) Clean image. (g) Base layer (RGB). (h) Detail layer (RGB). (i) Our de-rained (h). (j) Our de-rained result.

F. Impact of the Selected Low-Pass Filter

Though we choose the guided filter [24] to separate the base and detail layers for training the CNN, we found that the framework of Figure 2 is effective using other low-pass filters as well. Figure 15 shows one example of a de-raining result using different low-pass filters: guided filtering [24], bilateral filtering [25] and rolling guidance filtering [26]. As can be seen, though the low and high frequency decomposition look significantly different, the de-raining result is qualitatively similar and the three SSIM values of the de-rained results are almost the same. DerainNet is able to recognize and remove rain as long as it is isolated to the detail layer.

The method proposed in [10] also applies this decomposition strategy using the bilateral filtering, but used in a different model. We make a comparison with method [10] using bilateral filtering for our CNN as well. To ensure the low-pass filter removes all the rain streaks, we change the default parameters of the bilateral filtering in [10]. Specifically, we change the window size from 5 to 15 and intensity-domain standard deviations from 0.1 to 1. The difference in filtering operations between our method and [10] is that method [10] implements the pre-processing in the Y channel of YUV color space, while our method implements it in the RGB color space. Figure 16 shows the both intermediate and final de-rained results. As can be seen, both methods isolate the rain to the high-pass portion for de-raining, but [10] fails to completely remove rain streaks in Figure 16(c), while our result in Figure 16(i) has considerably better success. The final results are shown in Figures 16(e) and (j).

G. Training Convergence and Testing Runtime

Training required approximately two days to run. In Figure 17 we show the training convergence as a function of the number of backpropagations. While training requires a nontrivial amount of computation time, DerainNet is able to process new images very efficiently compared with current state-of-the-art de-raining methods. Table IX shows the average running time for three different image sizes, each
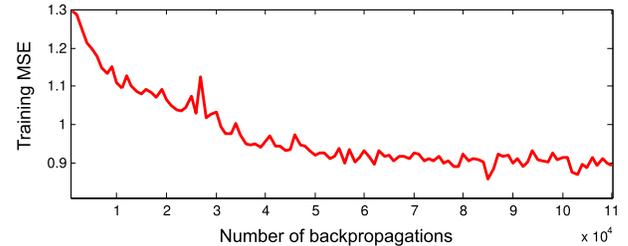

Fig. 17. The training convergence curve of DerainNet.

TABLE IX
COMPARISON OF RUNNING TIME (SECONDS)

Image size	Method [10]	Method [11]	Method [13]	Ours
250 × 250	68	53	196	**1.3**
500 × 500	76	230	942	**2.8**
750 × 750	99	782	1374	**5.4**

averaged over 10 testing images. (Note that these results do not factor in training time, but are for applying these methods to new data.) Since methods [10], [11] are based on dictionary learning and method [13] is based on Gaussian mixture model learning, complex optimizations are still required to de-rain new images, leading to a slower computation time. Our method has significantly faster running time since the testing procedure is completely feed-forward after network training. For even larger images, such as those taken by a typical camera, [10], [11], [13] take from several minutes to over an hour to process a new image, while our method requires roughly half a minute based on a parallel GPU implementation.

IV. CONCLUSION

We have presented a deep learning architecture called DerainNet for removing rain from individual images. Using a convolutional neural network on the high frequency detail content, our approach learns the mapping function between clean and rainy image detail layers. Since we do not possess the ground truth clean images corresponding to real-world

rainy images, we synthesize clean/rainy image pairs for network learning, and showed how this network still transfers well to real-world images. We showed that deep learning with convolutional neural networks, a technology widely used for high-level vision task, can also be exploited to successfully deal with natural images under bad weather conditions. We also showed that DerainNet noticeably outperforms other state-of-the-art methods with respect to image quality and computational efficiency. In addition, by using image processing domain knowledge, we were able to show that we do not need a very deep (or wide) network to perform this task.

REFERENCES

[1] K. Garg and S. K. Nayar, "Detection and removal of rain from videos," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun./Jul. 2004, pp. I-528–I-535.

[2] P. C. Barnum, S. Narasimhan, and T. Kanade, "Analysis of rain and snow in frequency space," *Int. J. Comput. Vis.*, vol. 86, nos. 2–3, pp. 256–274, Jan. 2010.

[3] J. Bossu, N. Hautiere, and J. P. Tarel, "Rain or snow detection in image sequences through use of a histogram of orientation of streaks," *Int. J. Comput. Vis.*, vol. 93, no. 3, pp. 348–367, Jul. 2011.

[4] V. Santhaseelan and V. K. Asari, "Utilizing local phase information to remove rain from video," *Int. J. Comput. Vis.*, vol. 112, no. 1, pp. 71–89, Mar. 2015.

[5] 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. (ICIP)*, Sep. 2013, pp. 914–917.

[6] 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. (ICCV)*, Dec. 2013, pp. 633–640.

[7] 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. (ICCV)*, Dec. 2013, pp. 1968–1975.

[8] D.-A. Huang, L.-W. Kang, M.-C. Yang, C.-W. Lin, and Y.-C. F. Wang, "Context-aware single image rain removal," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2012, pp. 164–169.

[9] 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.

[10] D.-A. Huang, L.-W. Kang, Y.-C. F. Wang, and C.-W. Lin, "Self-learning based image decomposition with applications to single image denoising," *IEEE Trans. Multimedia*, vol. 16, no. 1, pp. 83–93, Jan. 2014.

[11] 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.

[12] D.-Y. Chen, C.-C. Chen, and L.-W. Kang, "Visual depth guided color image rain streaks removal using sparse coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 8, pp. 1430–1455, Aug. 2014.

[13] 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.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[16] J. Xie, L. Xu, E. Chen, J. Xie, and L. Xu, "Image denoising and inpainting with deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 341–349.

[17] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[18] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1646–1654.

[19] L. Xu, J. S. J. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2014, pp. 1790–1798.

[20] J. S. J. Ren, L. Xu, Q. Yan, and W. Sun, "Shepard convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 901–909.

[21] L. Xu, J. Ren, Q. Yan, R. Liao, and J. Jia, "Deep edge-aware filters," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 1669–1678.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[23] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[24] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1397–1409, Jun. 2013.

[25] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. 6th Int. Conf. Comput. Vis. (ICCV)*, Jan. 1998, pp. 839–846.

[26] Q. Zhang, X. Shen, L. Xu, and J. Jia, "Rolling guidance filter," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2014, pp. 815–830.

[27] B. Gu, W. Li, M. Zhu, and M. Wang, "Local edge-preserving multiscale decomposition for high dynamic range image tone mapping," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 70–79, Jan. 2013.

[28] T. Qiu, A. Wang, N. Yu, and A. Song, "LLSURE: Local linear sure-based edge-preserving image filtering," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 80–90, Jan. 2013.

[29] G. Schaefer and M. Stich, "UCID: An uncompressed color image database," *Proc. SPIE*, vol. 5307, pp. 472–480, Dec. 2003.

[30] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.

[31] Y. Li, F. Guo, R. T. Tan, and M. S. Brown, "A contrast enhancement framework with JPEG artifacts suppression," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2014, pp. 174–188.

[32] 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.

[33] K. Garg and S. K. Nayar, "Photorealistic rendering of rain streaks," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 996–1002, Jul. 2006.

[34] A. K. Moorthy and A. C. Bovik, "A two-step framework for constructing blind image quality indices," *IEEE Signal Process. Lett.*, vol. 17, no. 5, pp. 513–516, May 2010.

Xueyang Fu received the master's degree in electronics and communication engineering from Xiamen University, Xiamen, China, in 2014, where he is currently pursuing the Ph.D. degree in signal and information processing. He is currently conducting the Ph.D. research as a Visiting Student with Columbia University, which is sponsored by the China Scholarship Council. His research interests include image processing, deep learning, and machine learning.

Jiabin Huang received the bachelor's degree in electronic information engineering from the Hubei University of Economics in 2015. He is currently pursuing the master's degree in electronic and communication engineering with Xiamen University, Xiamen, China. His research interests include image processing, deep learning, and machine learning.

Xinghao Ding 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 Post-Doctoral 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, sparse signal representation, and image processing.

Yinghao Liao received the B.S. and M.S. degrees in electronic engineering from Xiamen University, in 1982 and 1988, respectively. He is currently an Associate Professor with the School of Information Science and Engineering, Xiamen University, China. His research interests include image processing, sparse signal representation, and machine learning.

John Paisley received the B.S., M.S., and Ph.D. degrees in electrical engineering from Duke University. He was a Post-Doctoral Researcher with the Computer Science Department, UC Berkeley and Princeton University. He is currently an Assistant Professor with the Department of Electrical Engineering, Columbia University, and also a member of the Data Science Institute, Columbia University. His current research is machine learning, focusing on models and inference techniques for text and image processing applications.