Removing rain from single images via a deep detail network

Xueyang Fu[1] Jiabin Huang[1] Delu Zeng[2] Yue Huang[1] Xinghao Ding[1*] John Paisley[3]

[1]Key Laboratory of Underwater Acoustic Communication and Marine Information Technology, Ministry of Education & School of Information Science and Engineering, Xiamen University, China

[2]School of Mathematics, South China University of Technology, China

[3]Department of Electrical Engineering & Data Science Institute, Columbia University, USA

Abstract

We propose a new deep network architecture for removing rain streaks from individual images based on the deep convolutional neural network (CNN). Inspired by the deep residual network (ResNet) that simplifies the learning process by changing the mapping form, we propose a deep detail network to directly reduce the mapping range from input to output, which makes the learning process easier. To further improve the de-rained result, we use a priori image domain knowledge by focusing on high frequency detail during training, which removes background interference and focuses the model on the structure of rain in images. This demonstrates that a deep architecture not only has benefits for high-level vision tasks but also can be used to solve low-level imaging problems. Though we train the network on synthetic data, we find that the learned network generalizes well to real-world test images. Experiments show that the proposed method significantly outperforms state-of-the-art methods on both synthetic and real-world images in terms of both qualitative and quantitative measures. We discuss applications of this structure to denoising and JPEG artifact reduction at the end of the paper.

1. Introduction

Under rainy conditions, the impact of rain streaks on images and video is often undesirable. In addition to a subjective degradation, the effects of rain can also severely affect the performance of outdoor vision systems, such as surveillance systems. Effective methods for removing rain streaks are needed for a wide range of practical applications. However, when an object's structure and orientation is similar with that of rain streaks, it is hard to simultaneously remove rain and preserve structure. To address this difficult problem, we develop an end-to-end deep network architecture for removing rain from individual images. Figure 1 shows an example of a real-world test image and our result. To date, many methods have been proposed for removing rain from images. These methods fall into two categories: video-based methods and single-image based methods. We briefly review these approaches and then discuss the contributions of our proposed framework.

Figure 1: An example real-world rainy image and our result.

1.1. Related work

For video-based methods, rain can be more easily identified and removed using inter-frame information [3,4,10,21, 28, 35]. 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. This task is significantly more challenging since much less information is available for detecting and removing rain. Single-image based methods have been proposed to deal with this challenging problem. For example, in [20] kernel regression and a non-local mean filtering are used to detect and remove rain streaks. In [6], the authors propose a generalized model in which additive rain is assumed to be low rank. In general, however, success has been less noticeable than in video-based algorithms and there is still much room

*Corresponding author: dxh@xmu.edu.cn This work was supported in part by the National Natural Science Foundation of China grants 61571382, 81671766, 61571005, 81671674, U1605252, 61671309 and 81301278, Guangdong Natural Science Foundation grant 2015A030313007, Fundamental Research Funds for the Central Universities grants 20720160075 and 20720150169, and the CCF-Tencent research fund. X. Fu conducted portions of this work at Columbia University under China Scholarship Council grant No. [2016]3100.

1063-6919/17 $31.00 © 2017 IEEE
DOI 10.1109/CVPR.2017.186

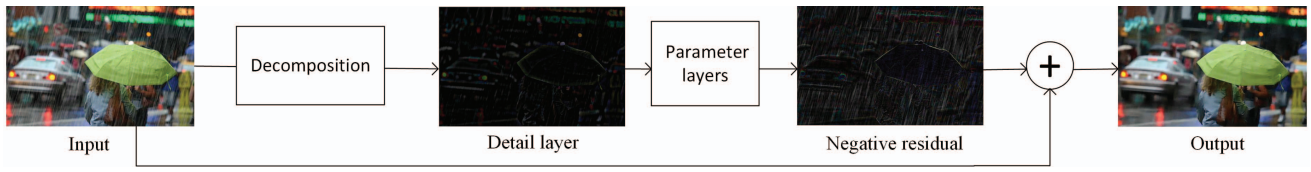


Figure 2: The proposed framework for single-image rain removal. Middle images show absolute value for better visualization.

for improvement.

Several methods using patch-based modeling have also been proposed and represent the current state-of-the-art [5, 14, 15, 18, 24, 25, 31]. For example, in [25] the authors use discriminative sparse coding to recover a clean image from a rainy image. Recently, [24] proposed a method based on Gaussian mixture models in which patch-based priors are used for both a clean layer and a rain layer. The authors show how multiple orientations and scales of rain streaks can be accounted for by such pre-trained Gaussian mixture models.

1.2. Our contributions

As mentioned, removing rain from a single image is significantly more difficult than from video. This is because most existing methods separate rain streaks from images using low-level image features [14, 18, 24, 25]. When an object's structures and orientations are similar with that of rain streaks, these methods have a difficult time simultaneously removing rain and preserving structure. Towards fixing this problem, we design a rain-removal method based on the convolutional neural network (CNN) [22, 23]. The deep CNN has not only achieved success on high-level vision tasks [12, 13] but has also been extended to problems such as image denoising [34, 36], super-resolution [7, 19], reducing artifacts of compression [8], image inpainting [26] and image dehazing [27]. CNNs are effective at increasing the ability of a model to explore and capture a variety of image characteristics [30].

In this paper, we design a new network architecture for single-image rain removal. As shown by the deep residual network (ResNet) [12], directly reducing the mapping range from input to output can make the learning process significantly easier. Based on this observation, we propose a "deep detail network" for removing the naturally high frequency rain content and evaluate it against state-of-the-art methods on both synthetic and real-world rainy images. The contributions of this work are summarized as follows:

1. We use a lossless "negative residual mapping," which we abbreviate as "neg-mapping", defined to be the difference between clean and rainy images. This is motivated by the observation that predicting the residual can significantly reduce the mapping range, which makes the learning process much easier for deep models.

2. We adopt the ResNet structure [12] as the parameter layers for a deep exploration of image characteristics. Rather than directly apply ResNet on the images, we exploit *a priori* knowledge and use the image detail layer as the input. Since this removes background interference, most pixel values in the detail layer are very close to zero. We find that this sparsity further improves the de-raining quality. Building on this observation, we argue that with an appropriate network design that avoids gradient vanishing, a deeper structure actually improves results for de-raining. This is in contrast to a common assumption that a deeper structure is not good for low-level image tasks.

3. To learn the network, we create and use a synthetic dataset of 14,000 rainy/clean image pairs. To our knowledge, this is significantly larger than previous de-raining data sets, since existing dictionary learning algorithms typically employ inefficient learning algorithms. Although the network is trained on synthetic rain data, we find that it generalizes very well to real-world rainy images for which these pairs are unavailable.

2. A Deep Detail Network

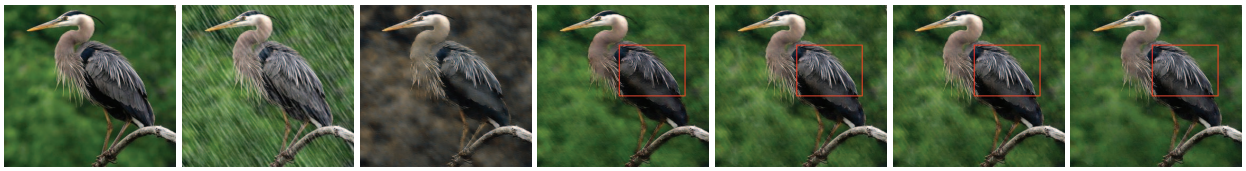
We illustrate the proposed de-raining framework in Figure 2. As discussed below, we define the detail layer and the negative residual to be the input and output of intermediate parameter layers. Because we lack access to the ground truth for real-world rainy images, we synthesize a dataset of rainy images using clean images to train the network. After training, the network can be used to output de-rained versions of an input rainy image.

2.1. Network design

We denote the input rainy image and corresponding clean image as $\mathbf{X}$ and $\mathbf{Y}$, respectively. Intuitively, a goal may be to directly train a deep CNN architecture $h(\mathbf{X})$ on multiple images to minimize the objective function

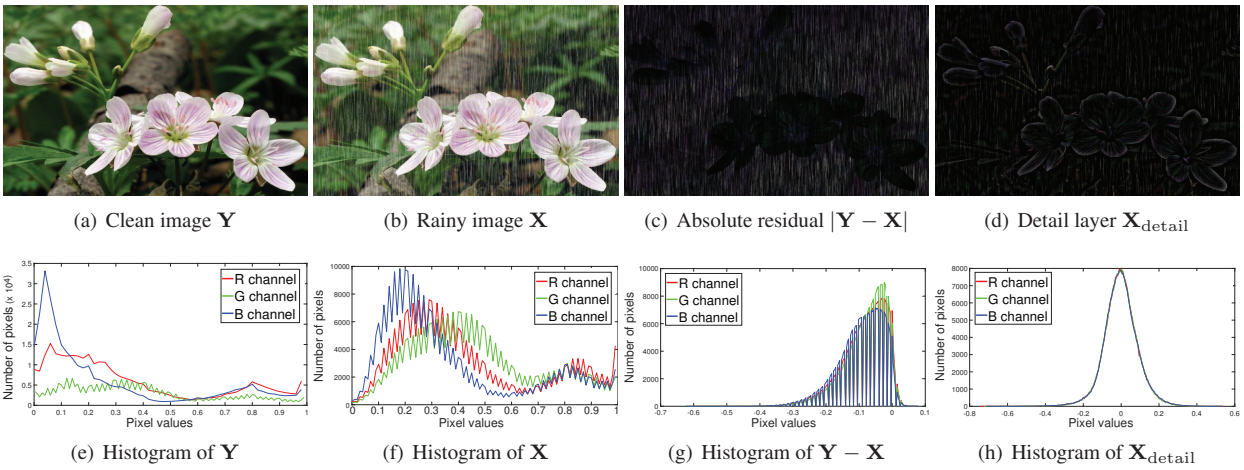
$$\mathcal{L} = \sum_i \|h(\mathbf{X}_i) - \mathbf{Y}_i\|_F^2, \quad (1)$$

where F is the Frobenius norm. However, we found that the result obtained by directly training on the image domain ("direct network") is not satisfactory. In Figure 3(b), we show an example of a synthetic rainy image which is used



(a) Ground truth (b) Rainy image (c) Direct network (d) ResNet (e) Neg-mapping (f) Neg-map+ResNet (g) Final network

Figure 3: De-rained results of different network structures. SSIM of (b)–(g) are 0.774, 0.490, 0.926, 0.936, 0.938 and **0.940**, respectively. White feathers on the bird's back are best preserved by our networks (e)–(f). All network depths are set to 26.



(a) Clean image $\mathbf{Y}$ (b) Rainy image $\mathbf{X}$ (c) Absolute residual $|\mathbf{Y}-\mathbf{X}|$ (d) Detail layer $\mathbf{X}_{\text{detail}}$

(e) Histogram of $\mathbf{Y}$ (f) Histogram of $\mathbf{X}$ (g) Histogram of $\mathbf{Y}-\mathbf{X}$ (h) Histogram of $\mathbf{X}_{\text{detail}}$

Figure 4: Range reduction and sparsity of the negative residual and detail layer. We show $|\mathbf{Y}-\mathbf{X}|$ for better visualization.

in the training process. In Figure 3(c) we see that, even though this image is used as a training sample, the de-rained image output by a CNN has a severe color shift. In general, we observed that the learned regression function underfits when directly trained on the image domain. To briefly explain this phenomenon: For simplicity, assume that images $\mathbf{X}$ and $\mathbf{Y}$ are normalized to between $[0,1]$ and have D pixels. Learning a regression function between $\mathbf{X}$ and $\mathbf{Y}$ maps from $[0,1]^D$ to $[0,1]^D$. In other words, the mapping range covers all possible pixel values, which makes it hard to learn the regression function well. Moreover, training a deep network directly on images suffers from gradient vanishing [12] even when using regularization methods such as batch normalization [16]. (As shown in Figure 10 in our experiments.) We will return to this analysis after presenting our proposed development.

2.1.1 Negative residual mapping

To improve the network learning process, it is important to reduce the solution space by compressing the mapping range [12]. As shown in Figure 4(e) and (g), we observe that when compared to the clean image $\mathbf{Y}$, the residual of the rainy image $\mathbf{Y}-\mathbf{X}$ has a significant range reduction in pixel values. This implies that the residual can be introduced into the network to help learn the mapping. Thus we use the residual as the output of the parameter layers, as shown in Figure 2. This skip connection can also directly propagate lossless information through the entire network, which is useful for estimating the final de-rained image. Because rain tends to appear in images as white streaks, most values of $\mathbf{Y}-\mathbf{X}$ tend to be negative, as shown in Figure 4(g). Thus we refer to this as "negative residual mapping" (*neg-mapping* for short). A modified objective that incorporates this idea can be written

$$\mathcal{L} = \textstyle\sum_i \|h(\mathbf{X}_i) + \mathbf{X}_i - \mathbf{Y}_i\|_F^2. \tag{2}$$

Figure 3(d) shows the de-raining result by the popular ResNet model [12], which we have modified to be suitable for image regression problems (as shown in Figure 5). The original ResNet structure removes rain streaks, but it also blurs the bird's feathers. In contrast, as shown in Figure 3(e) both color and object details can be preserved by simply embedding the neg-mapping without using the ResNet

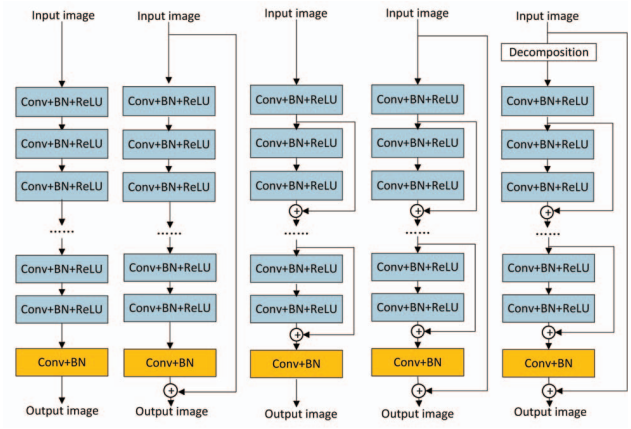


Figure 5: The five network architectures we considered for the rain removal problem. (Left to right) direct network, Neg-mapping, ResNet, ResNet+Neg-mapping and the final deep detail network that gave the best performance.

structure (also see Figure 5). This is because the lossless information is directly propagated and used to update parameters by the skip connection. Moreover, neg-mapping has a lower training and testing error than ResNet, as shown in our experiments in Figure 10. Though Figure 3(e) has a higher structure similarity index (SSIM) [33] than ResNet, as indicated in the caption, slight rain streaks remain in the output. This implies that rain streaks and object details are not totally distinguished by the model. This motivates the following developments of the neg-mapping idea.

2.1.2 Deep detail network

Since deeper architectures can increase the flexibility and capacity for exploring and modeling image characteristics [30], we use the ResNet [12] structure with neg-mapping to better distinguish rain streaks from object details. This structure guarantees that the input information can be propagated through all parameter layers, which helps to train the network [12]. However, we observe in Figure 3(f) that, even we if combine these two, slight rain streaks still remain in the output. Thus, in contrast to the original ResNet approach, we use the detail layer as the input to the parameter layers. To this end, we first model the rainy image as

$$\mathbf{X} = \mathbf{X}_{\text{detail}} + \mathbf{X}_{\text{base}}, \tag{3}$$

where the subscript 'detail' denotes the detail layer, and 'base' denotes the base layer. The base layer can be obtained using low-pass filtering of $\mathbf{X}$ [11, 32, 37] after which the detail layer $\mathbf{X}_{\text{detail}} = \mathbf{X} - \mathbf{X}_{\text{base}}$. After subtracting the base layer from the image, the interference of background is removed and only rain streaks and object structures remain in the detail layer. This is shown in Figure 4(d) and (h). We see that the detail layer is more sparse than the image since most regions in the detail layer are close to zero. Sparsity of the detail layer has been exploited in existing de-rain methods [14, 18, 25], though not in a deep learning framework.

Moreover, as shown in Figure 4(c),(d),(g) and (h), both the detail layer and the neg-mapping residual exhibit significant range reduction. Therefore, the *effective* mapping in this example is from smaller subsets of $[0,1]^D$ to $[0,1]^D$. This indicates that the solution space has shrunk and so network performance should be improved [12].

This motivates us to combine the detail layer $\mathbf{X}_{\text{detail}}$ with the proposed neg-mapping $\mathbf{Y} - \mathbf{X}$ as the input to the parameter layers of ResNet. Since we train the network on the detail layer, we refer to this as a "deep detail network." Figure 3(g) shows the final result by combing the proposed deep detail network with neg-mapping (see also Figure 5). Compared with other network structures, our final result not only achieves a good SSIM and convergence rate (Figure 10), but also has a cleaner visual de-raining effect.

2.2. Objective function and network architecture

The input of our de-rain system is a rainy image $\mathbf{X}$ and the output is an approximation to the clean image $\mathbf{Y}$. Based on the previous discussion, we define the objective function to be,

$$\mathcal{L} = \sum_{i=1}^{N} \|f(\mathbf{X}_{i,\text{detail}}, \mathbf{W}, \mathbf{b}) + \mathbf{X}_i - \mathbf{Y}_i\|_F^2, \tag{4}$$

where N is the number of training images, $f(\cdot)$ is ResNet, $\mathbf{W}$ and $\mathbf{b}$ are network parameters that need to be learned. For $\mathbf{X}_{\text{detail}}$, we first use guided filtering [11] as a low-pass filter to split $\mathbf{X}$ into base and detail layers. (Other approaches were also effective in our experiments.)

Removing image indexing, our basic network structure can be expressed as,

$$\begin{aligned}
&\mathbf{X}^0_{\text{detail}} = \mathbf{X} - \mathbf{X}_{\text{base}}, \\
&\mathbf{X}^1_{\text{detail}} = \sigma(\text{BN}(\mathbf{W}^1 * \mathbf{X}^0_{\text{detail}} + \mathbf{b}^1)), \\
&\mathbf{X}^{2l}_{\text{detail}} = \sigma(\text{BN}(\mathbf{W}^{2l} * \mathbf{X}^{2l-1}_{\text{detail}} + \mathbf{b}^{2l})), \\
&\mathbf{X}^{2l+1}_{\text{detail}} = \sigma(\text{BN}(\mathbf{W}^{2l+1} * \mathbf{X}^{2l}_{\text{detail}} + \mathbf{b}^{2l+1})) + \mathbf{X}^{2l-1}_{\text{detail}}, \\
&\mathbf{Y}_{\text{approx}} = \text{BN}(\mathbf{W}^L * \mathbf{X}^{L-1}_{\text{detail}} + \mathbf{b}^L) + \mathbf{X},
\end{aligned} \tag{5}$$

where $l = 1, ..., \frac{L-2}{2}$ with L the total number of layers, $*$ indicates the convolution operation, $\mathbf{W}$ contains weights and $\mathbf{b}$ biases, $\text{BN}(\cdot)$ indicates batch normalization to alleviate internal covariate shift [16], $\sigma(\cdot)$ is a Rectified Linear Unit (ReLU) [22] for non-linearity. In our network, all pooling operations are removed to preserve spatial information.

For the first layer, we use filters of size $c \times s_1 \times s_1 \times a_1$ to generate a_1 feature maps; s represents filter size and c represents the number of image channels, e.g., $c = 1$ for gray-scale and $c = 3$ for color image. For layers 2 through

(a) Ground truth (b) Rainy images (c) Method [25] (d) Method [24] (e) Our results

Figure 6: Three synthetic images with different orientations and magnitudes: (top-to-bottom) "girl", "flower", "umbrella".

$L-1$, filters are size $a_1 \times s_2 \times s_2 \times a_2$. For the last layer, we use filters of size $a_2 \times s_3 \times s_3 \times c$ to estimate the negative residual. The de-rained image is obtained by directly adding the estimated residual to the rainy image $\mathbf{X}$.

Discussion. It is worth noting that though the structure of our layer parameters is based on the original ResNet [12], as shown in Figure 5, the two networks have different motivations and implementations. ResNet was designed to simplify the learning process by changing the mapping form, while our deep detail network directly reduces the mapping range *a priori* using neg-mapping. ResNet aims to train an extremely deep neural network for high-level vision tasks, such as image classification and object detection. Our deep network uses image domain knowledge to focus on image regression problems and improving image quality. Where ResNet uses the image itself as the input to the network, our network uses the detail layer for improved training, as well as the proposed neg-mapping.

2.3. Training

We use stochastic gradient descent (SGD) to minimize the objective function in Equation (4). Since it is hard to obtain a large number of clean/rainy image pairs from real-world data, we synthesize rainy images using [1]. We collect 1,000 clean images from the UCID dataset [29], the BSD dataset [2] and Google image search to synthesize rainy images. Each clean image was used to generate 14 rainy images with different streak orientations and magnitudes. Thus, we train on a dataset containing 14,000 pairs of rainy/clean images. We randomly selected 9,100 images from which we generated three million 64×64 rainy/clean patch pairs. After training, the remaining 4,900 image pairs are used to evaluate the trained network.

3. Experiments

To evaluate our deep detail network, we use both synthetic and real-world data and compare with two recent state-of-the-art de-raining methods, one based on dictionary learning [25] the other on mixture modeling [24]. We use the Caffe software package to train our model [17].[1] It took approximately 14 hours to train our network using the settings below.

3.1. Parameter settings

We set the detail network depth to $L = 26$, and use SGD with weight decay of 10^{-10}, momentum of 0.9 and a mini-batch size of 20. We start with a learning rate of 0.1, dividing it by 10 at 100K and 200K iterations, and terminate training at 210K iterations. We set the filter sizes $s_1 = s_2 = s_3 = 3$ and filter numbers $a_1 = a_2 = 16$, both of which are described in Section 2.2. Larger filter sizes are able to capture more structural information and potentially improve results at the cost of algorithm speed; during our experiments, we found that 3×3 filter size generate results that are representative of our deep network structure, while still being computationally efficient. Since the process is applied on color images, we set $c = 3$, also described in Section 2.2. The radius of the guided filter for low-pass filtering is 15.

3.2. Results on synthetic test data

Figure 6 shows visual comparisons for three synthesized rainy images used for testing. As can be seen in the

[1]We share our Matlab implementation and synthetic rainy image dataset at: `http://smartdsp.xmu.edu.cn/cvpr2017.html`

Table 1: Quantitative measurement results using SSIM on synthesized test images.

Images	Ground truth	Rainy image	Method [25]	Method [24]	Ours
girl	1	0.65	0.71	0.80	**0.90**
flower	1	0.69	0.77	0.81	**0.92**
umbrella	1	0.75	0.80	0.82	**0.86**
4,900 test images	1	0.78 ± 0.12	0.83 ± 0.09	0.87 ± 0.07	$\mathbf{0.90 \pm 0.05}$

(a) Rainy images (b) Method [25] (c) Method [24] (d) Ours

Figure 7: Results on a real-world rainy image "street" and zoomed in region.

(a) Rainy images (b) Method [25] (c) Method [24] (d) Ours

Figure 8: Two results on real-world rainy images, (top) "people" and (bottom) "car."

third and forth columns, method [25] leaves significant rain streaks and method [24] contains rain artifacts. This is because methods [24, 25] only use low-level image features to remove rain. When an object's orientation and magnitude are similar with that of rain streaks, methods [24, 25] cannot efficiently distinguish objects from rain. In contrast, our deep CNN-based method shown in the last column can remove rain while still preserving details.

Since the ground truth is known for the synthetic problem, we use SSIM [33] for a quantitative evaluation. A higher SSIM indicates the de-rained image is closer to the ground truth in terms of image structure properties. (SSIM equals 1 for the ground truth.) As is evident in Table 1, the artifacts from methods [24, 25] result in a lower SSIM, in agreement with the visual effect. Also shown in Table 1 is the quantitative performance on 4,900 testing images constructed as discussed in Section 2.3.

3.3. Results on real-world test data

Figures 7 and 8 show three qualitative results on real-world images in varying proximity to objects of interest. Qualitatively, the proposed method shows the best visual

(a) Rainy images (b) De-hazed (a) (c) De-rained (b)

Figure 9: An example of heavy rain removal with dehazing.

performance on removing rain. Method [25] still contains rain streaks while method [24] generates over-smoothed results. Since no standard high-quality reference-free quantitative measure exists, we only show qualitative results for real-world data.

When dealing with heavy rain images can become hazy. This haziness does not appear in the synthesized images used to train our network. In this case, we found that applying a dehazing method as pre-processing is useful. We show an example in Figure 9, where combining our deep network with a dehazing algorithm [27] can further improve the visual quality on real-world rainy images.

3.4. Running time on test images

Compared with other non-deep methods, our proposed approach can process new images very efficiently. Table 2 shows the average running time of processing a test image for three different image sizes, each averaged over 100 testing images. Methods [24, 25] are implemented using CPUs according to the provided source code, while our method is tested on both CPUs and GPUs. Since method [25] is based on dictionary learning and method [24] is based on Gaussian mixture model learning, complex optimizations are still required to de-rain test images, which accounts for the slower computation time. Our method has significantly faster running time for new images since it is entirely feed-forward after network training.

Table 2: Comparison of test running time (seconds).

Image size	[25]	[24]	Ours (CPU)	Ours (GPU)
250×250	54.9	169.6	1.9	0.2
500×500	189.3	674.8	6.3	0.3
750×750	383.9	1468.7	12.6	0.5

3.5. Convergence of different network structures

In Figure 10, we show the average per-image training and testing error values as a function of training iteration for the different deep network designs described in Figure 5. We note that the results shown thus far correspond to the last network in Figure 5, which was selected based on the previous discussion. Among these methods, we observe

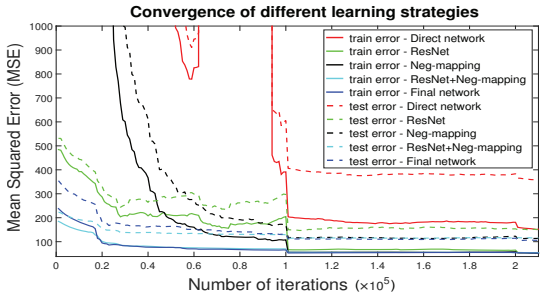


Figure 10: Convergence of different network structures. The drops indicated by vertical lines at 10^5 and 2×10^5 iterations are due to the scheduled learning rate division.

that the "direct network" and ResNet represent previous algorithms, while the remaining three represent options we considered. We recall that at 100K and 200K iterations we modify the step size by dividing by 10, which accounts for the jumps at these points.

We observe that our final network has among the best convergence rates on the training data (solid lines), while the testing performance is slightly better than the other two options we considered, and significantly better than the existing "off-the-shelf" deep learning approaches (dotted lines). We believe this improved performance is because we reduce the mapping range using neg-mapping in combination with the detail layer, which benefits the regression problem as discussed in Section 2. Moreover, our final network has a slightly cleaner visual de-raining effect than the other network structures, as previously shown in Figure 3.

3.6. Network depth versus breadth

It is commonly believed that deeper structures are not good for low-level image tasks [7, 27]. However, we argued above that, with an appropriate network design that incorporates domain knowledge, deeper structures can help to increase the network's modeling capacity and improve the de-raining result. This increased capacity can come in two forms: one is to increase the depth of the network by stacking more hidden layers, and the other is to increase the breadth of network by using more filters in each hidden layer. In this section, we test the impact of network depth and breadth on 100 synthetic rainy images.

Specifically, we test for depth $L =\in \{14, 26, 50\}$ and filter numbers $a_1 = a_2 \in \{16, 32, 64\}$; we show these results

Table 3: Average SSIM using different network sizes.

	$a_1 = a_2 = 16$	$a_1 = a_2 = 32$	$a_1 = a_2 = 64$
$L = 14$	0.906	0.912	0.915
$L = 26$	0.916	0.920	0.920
$L = 50$	0.921	0.926	0.928

Table 4: Average SSIM on 100 synthetic testing images. (Max std $= 0.032$, min std $= 0.026$)

depth	8	14	20	26	50
ResNet	0.896	0.904	0.909	0.907	0.917
Ours	0.896	0.906	0.915	0.916	0.921

(a) Ours 26|0.940 (b) ResNet 50|0.942 (c) ResNet 26|0.926

Figure 11: Deraining performance (#layers|SSIM)

in Table 3. As is clear, adding more hidden layers achieves better results over increasing the number of filters per layer. From this, we believe that with appropriate network design to avoid gradient vanishing, deeper structures can improve the modeling result. To balance the trade-off between performance and speed, we chose depth $L = 26$ and filter numbers $a_1 = a_2 = 16$ for our experiments above. We also compared performance of our network with ResNet as a function of depth. As shown in Table 4 and Figure 11, for deeper models, our network generates results that are comparable to ResNet using half the number of layers.

3.7. Comparison with deep learning based method

We also experimented with the deep learning approach designed for window dirt/drop removal [9], the only other related deep learning approach we are aware of for a related problem. As shown in Figure 12, the model in [9] does not compare well with our method. This is because the method developed in [9] corresponds to the direct network of Figure 5 using three layers. As motivated in Section 2, training directly on the image domain has drawbacks and doesn't perform well with deeper structures, hence the worse performance of [9]. By using negative mapping and pre-filtering the image, we showed how deeper structures can actually aid de-raining and improve performance.

3.8. Extension: Noise and JPEG artifacts reduction

Finally, we mention that our deep detail network can be directly applied to other kinds of degraded images. Figure 13 shows the experimental results of image denoising and JPEG compression artifact reduction. This test demonstrates that the proposed network is actually a general framework for image processing tasks.

(a) Rainy images (b) [9] (SSIM=0.78) (c) Ours (SSIM=0.86)

Figure 12: Comparison with deep learning method [9].

(a) Noise (top) and JPEG (bottom) (b) Our results

Figure 13: Denoising and reducing JPEG artifacts.

4. Conclusion

We have presented an end-to-end deep learning framework for removing rain from individual images. We showed that combining the high frequency detail layer content of an image and regressing on the negative residual information has benefits for de-raining performance, since it makes the training process easier by reducing the mapping range. Since we do not possess the ground truth clean images corresponding to real-world rainy images, we synthesized clean/rainy image pairs for network learning, and showed how the learned network still transfers well to real-world images. We showed that our method noticeably outperforms other state-of-the-art methods, based on dictionary learning and mixture modeling frameworks, with respect to image quality and computational efficiency.

As indicated in Section 3.8, the motivations for our "deep detail network" approach applies equally well to other problems, such as image denoising and JPEG artifact reduction. When the corruptions between the desired image and observed image are of high-frequency, training a network to map from the high-frequency input details to the high-frequency output differences significantly simplifies the deep learning problem. As we observed, this translates to significant improvement of the performance of the learned network on new images.

References

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

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(5):898–916, 2011.

[3] P. C. Barnum, S. Narasimhan, and T. Kanade. Analysis of rain and snow in frequency space. *International Journal of Computer Vision*, 86(2-3):256–274, 2010.

[4] J. Bossu, N. Hautiere, and J. Tarel. Rain or snow detection in image sequences through use of a histogram of orientation of streaks. *International Journal of Computer Vision*, 93(3):348–367, 2011.

[5] D. Y. Chen, C. C. Chen, and L. W. Kang. Visual depth guided color image rain streaks removal using sparse coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(8):1430–1455, 2014.

[6] Y. L. Chen and C. T. Hsu. A generalized low-rank appearance model for spatio-temporally correlated rain streaks. In *ICCV*, 2013.

[7] C. Dong, C. L. Chen, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016.

[8] C. Dong, Y. Deng, C. C. Loy, and X. Tang. Compression artifacts reduction by a deep convolutional network. In *ICCV*, 2015.

[9] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In *ICCV*, 2013.

[10] K. Garg and S. K. Nayar. Detection and removal of rain from videos. In *CVPR*, 2004.

[11] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409, 2013.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[14] 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 Transactions on Multimedia*, 16(1):83–93, 2014.

[15] D. A. Huang, L. W. Kang, M. C. Yang, C. W. Lin, and Y. C. F. Wang. Context-aware single image rain removal. In *IEEE ICME*, 2012.

[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[17] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM International Conference on Multimedia*, 2014.

[18] L. W. Kang, C. W. Lin, and Y. H. Fu. Automatic single image-based rain streaks removal via image decomposition. *IEEE Transactions on Image Processing*, 21(4):1742–1755, 2012.

[19] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016.

[20] J. H. Kim, C. Lee, J. Y. Sim, and C. S. Kim. Single-image deraining using an adaptive nonlocal means filter. In *IEEE ICIP*, 2013.

[21] J. H. Kim, J. Y. Sim, and C. S. Kim. Video deraining and desnowing using temporal correlation and low-rank matrix completion. *IEEE Transactions on Image Processing*, 24(9):2658–2670, 2015.

[22] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[24] Y. Li, R. T. Tan, X. Guo, J. Lu, and M. S. Brown. Rain streak removal using layer priors. In *CVPR*, 2016.

[25] Y. Luo, Y. Xu, and H. Ji. Removing rain from a single image via discriminative sparse coding. In *ICCV*, 2015.

[26] J. S. Ren, L. Xu, Q. Yan, and W. Sun. Shepard convolutional neural networks. In *NIPS*, 2015.

[27] W. Ren, S. Liu, H. Zhang, J. Pan, X. Cao, and M. H. Yang. Single image dehazing via multi-scale convolutional neural networks. In *ECCV*, 2016.

[28] V. Santhaseelan and V. K. Asari. Utilizing local phase information to remove rain from video. *International Journal of Computer Vision*, 112(1):71–89, 2015.

[29] G. Schaefer and M. Stich. UCID: An uncompressed color image database. In *Storage and Retrieval Methods and Applications for Multimedia*, 2003.

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[31] C.-H. Son and X.-P. Zhang. Rain removal via shrinkage of sparse codes and learned rain dictionary. In *IEEE ICME*, 2016.

[32] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998.

[33] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[34] J. Xie, L. Xu, E. Chen, J. Xie, and L. Xu. Image denoising and inpainting with deep neural networks. In *NIPS*, 2012.

[35] S. You, R. T. Tan, R. Kawakami, Y. Mukaigawa, and K. Ikeuchi. Adherent raindrop modeling, detection and removal in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1721–1733, 2016.

[36] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 2017. doi: 10.1109/TIP.2017.2662206.

[37] Q. Zhang, X. Shen, L. Xu, and J. Jia. Rolling guidance filter. In *ECCV*, 2014.