Recursive Multi-Scale Image Deraining With Sub-Pixel Convolution Based Feature Fusion and Context Aggregation

SUVASH SHARMA\textsuperscript{1,2}, (Student Member, IEEE), BO TANG\textsuperscript{2}, (Member, IEEE), JOHN E. BALL\textsuperscript{2}, (Senior Member, IEEE), DANIEL W. CARRUTH\textsuperscript{1}, AND LALITHA DABBIRU\textsuperscript{1}

\textsuperscript{1}Center for Advanced Vehicular Systems, Mississippi State University, Starkville, MS 39762, USA
\textsuperscript{2}Department of Electrical and Computer Engineering, Mississippi State University, Starkville, MS 39762, USA

Corresponding author: Bo Tang (tang@ece.msstate.edu)

This work was partially supported by the Center for Advanced Vehicular System and the National Institute of Justice (NIJ) under Grant 2018-75-CX-K002.

ABSTRACT
Along with several other low-vision based computer vision problems, single image deraining is also taken as a challenging one due to its ill-posedness. Several algorithms based on convolutional neural networks are devised that are either too simple to provide acceptable deraining results due to under-deraining or have a complex architectures that may result into over-deraining. In this paper we propose a deraining algorithm that is capable of boosting the reconstruction/deraining quality without the problem of over or under-deraining. Along with the originally proposed network, two of its’ light-weight versions with reduced computational costs are also devised. Basically, we propose a recursively trained architecture that has two major components: a front-end module and a refinement module. The front-end module is based on dense fusion of lower label features followed by sub-pixel convolutions (pixel shuffling based convolutions). To refine and generate the enhanced deraining results further, we cascade a refinement module to the front-end module using multi-scale Context Aggregation Network (CAN) which includes feature fusion and pixel shuffling based convolutions. We present the deraining results in terms of Structural Similarity Index (SSIM) and Peak Signal to Noise Ratio (PSNR) on several benchmarks and compare with current state-of-the-art algorithms. With comprehensive experiments on both real-world and synthetic datasets and extensive ablation study, we demonstrate that our approach produces better results compared to existing methods.

INDEX TERMS
Single image deraining, multi-scale context aggregation, sub-pixel convolution, feature fusion.

I. INTRODUCTION
Low-level vision problems, sometimes called inverse vision problems, have been receiving significant attentions since many image processing algorithms only achieve their best performance when the quality of the input image is high. For example, an application that identifies all players in a soccer field needs to first reconstruct all invisible parts due to motion blurring for reliable recognition. A number of techniques are proposed to mitigate adversarial effects in the low-level vision problem such as object blurring [1], [2], haze [3], [4], fog [5], rain [6]–[9], shadow [10], [11], etc. In this paper we specifically consider the problem caused by rain on the visibility of images. Rainy weather or rain droplets on the camera are common occurrences that have negative impacts for computer vision applications like autonomous driving, vision based robots, and surveillance systems. These effects create severe modifications in the image causing degradation in background information which need to be processed further so as to extract the original clean image. Such techniques are developed and devised under the name of rain-removal or deraining algorithms.

In the past, a number of rain removal techniques for video-based systems (or for sequential image data) were proposed which mainly used the inter-frame information for detection and removal of rain [12]–[15]. However, detection...
and removal of rain streaks from a single image is quite difficult compared to those of the video-based methods. These difficulties are mainly due to randomness in physical nature of rain streaks, ill-posedness of the problem, cumbersome finding of proper priors, and unavailability of exact ground truths [16]. However, ample amount of techniques either using sparse representations [17], [18] or deep learning-based methods [19]–[23] are gaining popularity.

With impressive results, several deep learning based deraining architectures have been proposed. In [19] and [24], a detailed layer is designed to filter out low frequency components and the network learns on sparse high frequency representation which subsequently helps to reduce the input-output mapping range and expedites the training process. In [20], a contextualized dilated network followed by rain streak, rain region, and background prediction networks are used. The network is optimized in a recurrent fashion by formulating a multitask optimization problem by jointly minimizing the Mean Absolute Error (MAE) with respect to each of the output branches. Generative Adversarial Network (GAN) based rain removal algorithms are proposed in [25] and [6]. In [25], a conditional GAN is proposed where the generator network generates the derained images and the discriminator network verifies whether its input is the generator’s output (fake image) or the clear image (ground truth). The two networks are adversarially optimized such that the generator generates the images that are close to the ground-truth and the discriminator makes the faulty classification of the generator’s output. In [6], similar GAN setting is used where the generator is fed with the concatenation of rainy image and the corresponding attention map. The attention map is supposed to provide special focus on the rain streaks while removing them. A multi-stream deraining network based on rain density classification is proposed in [26]. The label from the classification output is fused to a multi-stream densely connected network in order to achieve the density-aware deraining. A non-local neural networks [27] based encoder-decoder network is proposed in [21] to accurately learn the abstract features and rain streaks modeling. An iterative approach has been used in [22] and [9] for progressive deraining and enforcing the reduction in number of parameters due to parameter sharing.

Despite the success of the rain removal techniques, some of those are focused on producing high quality results with complex deep neural networks; however, these networks either under-derain or over-derain. For example, as specified in [26], [20] under-derains, i.e., leaves some rain scars, and [19] over-derains, i.e., removes or modifies some important information in the image. On the other hand, methods based on GANs, due to their inherent nature of training instability, are difficult to optimize as well as to find the precise hyperparameters. Not only in terms of networks, some algorithms are based on optimization of complex cost functions and their combinations [6], [25], [26], [20]. Such optimization tasks are comparatively costly and introduce the problem related to training convergence. In this work, we develop the algorithm with single cost function as an objective for optimization that can provide high quality deraining results without the problem of over-deraining and under-deraining.

The proposed network is based on the residual blocks [28] and the dense feature fusion [29]. Basically, the whole architecture can be divided into two parts: The first part, called the front-end module, is a combination of residual blocks and fusion blocks, which we describe later in section III in detail. The output of the previous residual blocks are fused densely within the fusion blocks before proceeding to later stages. Fusion block comprises two normal convolutional layers and one pixel-shuffle based convolution, also called sub-pixel convolution, layer that has been suggested in [30]. The pixel-shuffle based convolution is performed by combining the input channels such that they contribute in forming a high resolution feature map with reduced channel depth and successively applying convolution operations.

In [30], it has been shown that any feature map of size \((H \times W \times N)\) can be rearranged to the upscaled feature maps of size \(rH \times rW \times \frac{N}{r^2}\) without any loss of information. This property has important effect that the convolution in the upscaled feature domain has better modeling power compared to the standard resize convolutions with the same computational complexity [31], [32]. Furthermore, in [31] it is demonstrated that the reconstructed/generated images or any dense labels using sub-pixel convolution are free of checkerboard artifacts that are prevalent in the images upsampled with deconvolution. This property holds huge importance in case of low-level vision problem where the reconstructed images are expected to be of high-quality, and artifacts free. This technique has been proven successful for generating high quality super resolution images in [30]. Taking this as an inspiration, we also utilize the pixel-shuffle based convolutions in our work.

Furthermore, in order to properly handle variable-shaped rain streaks, we introduce a multi-scale convolutional module based on the inception network [33] at the entry of the front-end. The second part, called refinement module, is cascaded with the front-end module in order to further finetune the deraining output and generate high-quality enhanced results. This module is largely a multi-scale CAN. Our multi-scale CAN is 32 feature maps based basic architecture proposed in [34], which is prepended with the same multi-scale convolutional module as the front-end. The CAN module has been proven to produce enhanced images in [35] due to dilated convolutions which is beneficial in two ways for our network. First, it fine-tunes the partially derained images from the front-end module and produces enhanced results. Second, due to the dilated convolutions, it helps in preserving the global context eradicating the problem of over-deraining. We also use one fusion block at the output of CAN which fuses all the previous outputs of CAN layers. However, we separate fusion block from pixel-shuffle convolutional layer so as to make the use of residual connection after the fusion block.

Inspired from [9], [22], we use a recurrent learning scheme which allows the progressive unfolding of the network without increasing the model parameters; this approach ultimately
helps us to limit the computational cost in-terms of training even with the deeper architecture. The cascaded connection of our front-end module and multi-scale CAN module recursively learns the residual of the input image as it helps to reduce the mapping range and hence helps in training convergence. A single stage of our proposed network looks as shown in Fig. 1.

In the front-end module, even though we fuse the lower level feature maps densely in fusion blocks, we limit the output number of channels for each block to 32. In order to train the network, a human visual system based metric called Structural Similarity Index Metric (SSIM) rather than Euclidean distance based metrics is used. While doing so, we use the negative of SSIM as a cost function to formulate the image deraining as a minimization problem. In order to justify the importance of each of our module, especially the multi-scale inception modules and CAN module, we perform a detailed ablation study. Addition of each of these modules helps in performance improvement in terms of image quality. Extensive experiments are performed in order to explore the effect of recursion depth on the deraining image quality.

II. RELATED WORK

As a large number of techniques have been proposed for single image deraining task, reviewing all of them will be out of scope of this paper. However, in general, those techniques lie in two major categories: model-driven methods and data-driven methods. This section briefly surveys some relevant and competitive algorithms in these categories.

A. MODEL-DRIVEN METHODS

Model-driven methods form an optimization problem based on some priors such as background image priors and rain streaks priors. As specified in [16], the general model-based
optimization problem is formed as given by Eq. (1).

\[
\min_B \| I - B - S \|^2 + \gamma(B) + \psi(S) + \Omega(B, S), \tag{1}
\]

where \( I \), \( B \), and \( S \) are the rainy image, background image and rain streaks layer respectively. \( \Gamma(B) \), \( \psi(S) \), and \( \Omega(B, S) \) are the priors corresponding to background image, rain layer and joint prior to describe the relationship between them, respectively. This class of algorithms are mostly based on sparse coding methods and Gaussian Mixture Models (GMMs). Sparse coding methods are mainly based on rainy image decomposition into the basis vectors corresponding to background image and rain streak layer [36], [37]. GMM is applied in [38], which models rainy image as a mixture of rain, and background distribution and is based on total variation minimization.

### B. DATA-DRIVEN METHODS

Basically, learning based methods are data-driven methods. Recently, deep learning based methods are gaining much popularity for deraining. These methods are based on a direct mapping of the rainy image to a derained image in reference to clean, rain-free image. In [22] and [9], simple and efficient convolutional neural network based deraining methods are proposed that are trained in a recursive fashion. Input image is decomposed into low and high frequency components [19], where the learning is performed by mapping the high frequency components which correspond to rain streak information and important edge information in the background. This helps to reduce mapping range and supports training. In [23], a hierarchical representation of wavelet-based recurrent approach is used. This method is also based on separation of low and high frequency components where the low-frequency component is subjected to rain removal which is then used as a guiding image during the recovery process from high-frequency image. A non-local neural network based deraining method is used in [21] where each pixel in the next-level feature is contributed by all the pixels in lower level features. With new real-world rain dataset, [7] proposes a local to global contextual features based attention module that considers the direction variability in the rain streaks. In [39], it is hypothesized that the rain density and direction does not change drastically with the feature scales. The deraining architecture is proposed based on the effectiveness of rain-streak location information in a multiscale fashion rather than about the density information. A new approach based on the distortion level of each image patch at different locations using a confidence measure guided training technique is proposed in [40]. An encoder-decoder based deraining architecture is proposed in [41] where the decoder layers are conditioned on the embedding learned by a separate branch towards encoder side. For its effectiveness a multi-stage training approach is followed. In [42] and [43] variational autoencoder based approach is proposed so as to use its’ generative ability. A density estimation method in order to estimate the density map is used based on the fact that the rain density varies spatially and channel-wise. A multiscale representation of rainy images is explored and the feature similarity/redundancy of the prevailing rain-streaks among same and different scaled images is exploited in [44] to derain the images. Very few approaches are proposed considering the semi-supervised learning scheme which is very important in case of real-world image deraining due to the lack of paired rainy-clean images. Recently [45] and [46] propose a semi-supervised deraining techniques where the network is trained simultaneously with labelled and unlabelled data. In [45], for unsupervised loss, rain residual is modeled as a likelihood in a gaussian mixture model. In [46], gaussian process based non-parametric approach is used where the intermediate latent space from the network is used to generate the pseudo ground truth for the unlabeled data.

### III. PROPOSED METHOD

In this section, we describe the details of our proposed deraining architecture and our training technique. The proposed architecture consists of two modules: a front-end module and a refinement module. The rainy image input is fed to the front-end module through a basic inception-based module and is subjected to the rain removal with cascaded residual blocks and fusion blocks. As we train our network recursively, we use one layer of convolutional Long Short-Term Memory (LSTM) to incorporate the long-term dependencies among the features following the successful use of it in [6] and [9]. The derained output from the front-end module is directed to the refinement module which refines the output of front-end module and generates the enhanced output. We use the inception-based multi-scale module at the entry of the refinement module so as to focus on the remnant of variable sized rain streaks and removal of them, if any. The overall architecture is shown in Fig. 1. Inspired from [9], in our recursive training scheme, at each time stage \( t \), the output \( f(x_{t-1}) \) is concatenated with the original rainy input \( f \). If \( x_t \) represents the network input at the \( t^{th} \) recursion stage, then at any time stage \( t \), the following expression holds true:

\[
x_t = C(i, f(x_{t-1})) \tag{2}
\]

where \( C(i, j) \) represents the depth-wise concatenation of feature maps \( i \) and \( j \). Also, it is noted that \( f(x_{t-1}) = I \) for the initial case.

### A. FRONT-END MODULE

As shown in Fig. 1, our front-end module comprises of four different modules: inception-based multi-scale module, a general convolutional LSTM module with 32 feature maps, residual blocks, and fusion blocks. Within the front-end module, there occurs the sequential connections in each module item. However, each fusion block takes the output of all the fusion blocks in addition to the output of LSTM module. In the following subsections we give the brief overview of each module item.
1) INCEPTION-BASED MULTI-SCALE MODULE
Inspired from the success of multi-scale modules used in [20] and [26] for leveraging the size variability of the rain streaks, use of inception-based multi-scale module is proposed in this work. In [20], a three-stream and two-layer dilated convolution based multi-scale module, called the contextualized dilated network, is proposed in order to capture the context and size variability of the rain streaks. Using this multi-scale structure in our proposed architecture, both the deraining performance and the frame rate decreased as shown in Table 6 in section V. On the other hand, the multi-scale structure in [26] uses three streams where each of them consists six dense blocks with 3 × 3, 5 × 5, and 7 × 7 kernel sizes. The output of the dense blocks are further concatenated and passed through two convolutional layers. Compared to the proposed inception-based multi-scale structure, the one in [26] has higher computational cost due to several reasons like dense fusion and concatenation of the low level features in all the three branches, higher number of convolutional operations in each branch, and higher filter sizes. Also, such dense fusion of lower level feature-maps is used in our fusion block and hence it would potentially be redundant when used within our multi-scale structure.

As shown in Fig. 2, a single module of inception network-based [33] multi-scale structure is used where three different filters of size 1 × 1, 3 × 3, and 5 × 5 are available. In two branches we use 1 × 1 filters in order to control the output number of channels to the desired value (12 in our case) for the convolutional blocks that use the filters of size 3 × 3 and 5 × 5. Outputs of all three branches are concatenated to form a 32 (=8 + 12 + 12) channels feature map which is then fed to the LSTM module. Since we use the same inception-based multi-scale structure in refinement module, the number of input feature maps to it is either 6 or 32. As specified earlier in Section 2, at each recursive stage we concatenate the previous output and the original image and hence the input number of channels is 6. This multi-scale arrangement is used in order to capture the size variability of rain streaks in the input image effectively.

2) RESIDUAL BLOCK
The residual block is a simple combination of two 3 × 3 convolutions and a ReLU (Rectified Linear Unit) activation arranged alternately. The residual structure is as shown in the middle block in Fig. 2. The same structure of residual block is repetitively used in our architecture. This setup largely helps in reducing model parameters as well as helps in reducing mapping range while training.

3) FUSION BLOCK
Inspired by better feature propagation and feature reuse capabilities of [29], our fusion block aggregates all feature maps from all previous fusion blocks and the LSTM along with the output of preceding residual block. In total we use five fusion blocks where each nth block gets 32(n + 1) feature maps and generates 32 of them. As shown in third module in Fig. 2, this block is comprised of four different layers. The entry layer of this block is mostly dedicated to fuse dense features which is done using 1 × 1 convolution and a ReLU activation. In order to use the pixel-shuffle based convolution, we use one 3 × 3 convolution before it which increases the number of feature maps by four folds (=128). As explained in section I, the pixel shuffling rearranges the pixels such that the shape of the feature maps gets transformed from $H \times W \times N$ to $rH \times rW \times n$, where, $N$ and $n$ represent the total number of feature maps and $n = N/r^2$. So, the total number of feature maps gets reduced by $r^2$ times in exchange of increasing the resolution by $r^2$ times. So we increase the feature-maps number by four times before applying pixel shuffling to get the feature maps of double the resolution on each dimension. We then apply a 3 × 3 convolution on these high resolution feature maps. As we densely combine the features within the fusion blocks and train the network in recursive fashion, such shuffling of pixels also helps in exploiting the
long-term dependencies [47] across the fused features as well as across the recursive stages. After sub-pixel convolution, we downsample the high-resolution features using bilinear interpolation so as to preserve the original resolution.

**B. REFINEMENT MODULE**

A refinement module is further designed to remove any remnants of the rain-streaks in the feature maps from the front-end module and generate the enhanced results considering global context. As shown in Fig. 1, CAN occupies the bulk of our refinement module. The inception-based multi-scale module has the same purpose and structure as used in the front-end module. The only difference is that it takes 32 feature maps as input. We use one instance of fusion block in this module too. However, the sub-pixel convolution layer is separated from remaining layers within the fusion block so as to use the residual connection within the CAN module. This fusion block takes all the lower level feature maps from CAN layers that are concatenated across the depth. The following subsection gives the brief overview of CAN module.

1) **CONTEXT AGGREGATION NETWORK**

Since a CAN is capable of increasing the receptive field through dilated convolutions, which then helps in maintaining the local and global contexts [34], and its application in reconstruction of the enhanced images [35], we make the use of it in our work which helps to boost the deraining performance. A six-layer CAN structure is used in [22] with 24 feature maps at each layer recalibrated by squeeze and excitation [48] method. Our CAN module is a basic version from [34] with 32 feature maps. Each layer in CAN is a combination of eight dilated convolutions and ReLU activation. The dilation rates used are 1, 2, 4, 8, 16, 16, 1, and 1.

**C. LIGHT-WEIGHT MODULES**

In this section we describe variants of the residual block and fusion block that help the deraining network to be lightweight. We call our proposed network with these modules as light-weight versions. Our original residual block is simply a cascaded combination of two convolutions and two relu activations in the form of \((\text{conv} + \text{relu} + \text{conv} + \text{relu})\) with a residual connection from input to output. In order to reduce the number of trainable parameters with possibly a very small drop in deraining quality, the depthwise convolutions in the residual block are replaced with group convolutions [49] to form efficient residual block (residual-E block) [50]. While doing so we keep the group size equal to two. At the output the residual-E block consists a \(1 \times 1\) convolutional layer. Mathematically, for a kernel of size \(K\) and convolution group size \(G\), the computational cost of residual-E block is \(2GK^2/(2K^2 + G)\) times lower compared to the original residual block [50]. Inspired from this fact, we replaced all five residual blocks by residual-E blocks and form a new deraining network which we called the light-weight version-1.

In light-weight version-2, we incorporate such group convolutions in first and second convolutional layers within fusion block too. During this time, we use the convolution group of two in the first convolutional layer and group of four in the second convolutional layer that lies before pixelshuffle block. Such changes in fusion block are made along with the use of residual-E blocks in place of residual blocks in the front-end module.

**D. TRAINING**

Inspired from [9], we use the negative of SSIM as our learning objective. Different from the loss functions like \(l_1\) and \(l_2\) losses which are based on euclidean distances, SSIM is a perceptual metric based on the image properties like luminance, contrast, and structure. Mathematically, SSIM is defined as in Eq.3 [51]:

\[
SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{\mu_x^2 + \mu_y^2 + C_1^2 + C_2^2}
\]

where \(x\) and \(y\) are the two images under comparison, \(\mu_x\), \(\mu_y\), \(\sigma_x\), and \(\sigma_y\) are the means and standard deviations of corresponding images, respectively, and \(C_1\) and \(C_2\) are the small constants used to avoid instability when \(\mu_x^2 + \mu_y^2\) is very close to zero. Using the negative of \(SSIM(x, y)\) as a cost function to be minimized, we start training with a learning rate of 0.001 with a multi-step learning schedule using the Adam optimizer [52]. While training for 100 epochs, the learning rate is decreased by a factor of 0.1 at the 33rd and 66th epochs. We use a batch size of four and a single Nvidia Quadro GP100 GPU with 16G memory. Furthermore, We implemented and analyzed our network using Pytorch-0.4.1 [53] library in python3.6 and ubuntu-16.04 environment.

**IV. RESULTS**

In this section we first report our quantitative and qualitative results on several datasets. We also compare the results of our network to that of several other state-of-the-arts methods to show the superiority of our method.

**A. QUANTITATIVE AND QUALITATIVE RESULTS ON SYNTHETIC DATASET**

We use four different publicly available benchmark datasets: DDN-Dataset [19], Rain100L [20], Rain100H [20], and Rain12 [38]. These datasets contain a mixture of low to high rain density and various natural properties of the rain-streaks including their random shapes and distributions. For all the datasets, we train our model from scratch. As in [9], since Rain12 only has 12 images, the trained model on Rain100L is directly applied to test the performance on these images. For the DDN-dataset, we ran our network with a recursion depth of two because this dataset has high volume and it takes a long time when the recursion depth is set to five. So, we just calculate the metrics after the recursion depth of two and we still get the enhanced performance where it surpasses the result of [9] by 0.1dB PSNR and 0.4% of SSIM. This result would improve further when recursion depth of five is used as suggested by the result expressed in Table 4. For Rain100L, Rain100H, and Rain12, we achieve 0.66dB, 1.58dB, and
TABLE 1. Quantitative comparison of various state-of-the-art deraining algorithms on different synthetic datasets in-terms of PSNR/SSIM. “−” represents that the corresponding paper does not provide the code or is too computationally heavy to run in our environment. The best result for each dataset is colored with blue. Our proposed network exceeds all the state-of-the-art methods in terms of both the PSNR and SSIM metrics. Our presented results have a recursion depth value of five. Note that the information below each algorithm’s name on the top row is the publication and year. Furthermore “*” represents that our network is trained with the recursion depth of two to retrieve the result fast as the training with corresponding dataset takes very long time with recursion depth of five in our single GPU environment due to its’ high volume.

| Dataset    | GMM [38] (CVPR’16) | DDN [19] (CVPR’17) | JORDER [20] (CVPR’17) | DID-MDN [26] (CVPR’18) | NLEDN [21] (MM’18) | RWL [23] (TIP’19) | PRENET [9] (CVPR’19) | Ours       |
|------------|---------------------|--------------------|-----------------------|-------------------------|---------------------|-------------------|---------------------|------------|
| DDN-Dataset| 25.64/0.836         | 28.24/0.8654       | 28.72/0.874           | 26.17/0.840             | 29.79/0.897         | -                 | 32.60/0.946        | 32.78/0.950* |
| Rain100L   | 28.25/0.876         | 25.99/0.814        | 35.23/0.967           | 30.48/0.932             | 36.57/0.974         | 36.75/0.975       | 37.48/0.979        | 38.14/0.982 |
| Rain100H   | 15.96/0.418         | 23.93/0.7430       | 26.54/0.835           | 26.35/0.828             | 30.38/0.893         | 26.89/0.840       | 29.46/0.899        | 31.07/0.915 |
| Rain12     | 30.02/0.855         | 31.78/0.900        | 33.92/0.953           | 30.14/0.876             | -                   | 36.32/0.963       | 36.69/0.962        | 36.86/0.965 |

FIGURE 3. Comparative demonstration of visual quality by different models in Rain100H [20] dataset. Compared to the other state-of-the-art methods, our proposed network has better reconstruction quality. When zoomed in, the circled area in the figure shows the improvement in that specific region compared to the previous method.

FIGURE 4. Comparative demonstration of visual quality by different models in Rain100L [20] dataset. Compared to the other state-of-the-art methods, our proposed network has better reconstruction quality as others leave several rain-streaks as a result of under-deraining. This can be verified when zoomed in and observed over the image area.

0.17dB performance gain in-terms of PSNR over the method proposed in [9], which exhibits similar training scenario to ours. Similarly, 0.3%, 2.1%, and 0.1% performance gain was achieved in-terms of SSIM. Rain100H is high density rain-streak-based dataset compared to Rain100L and Rain12. Our method is proven to be more effective when the rain streaks in the images have higher density. As shown in Table 1, the performance achieved by our proposed network is better compared to other famous methods too.

In Figure 3, we show the qualitative results of various state-of-the-art methods along with the results generated by our network on Rain100H [20]. We can see there are a-lot-of scars of the rain-streaks in the images from [20]. These rain-streaks are improved in case of [9], however we can still notice the smoothing effects. Such smoothing effect is greatly eradicated in the images generated by our method. When zoomed-in around the blue circled area of the images in Figure 3, we can see smoothing effect and/or artifacts introduced by other three methods are eradicated by our proposed method. We can see that our method is successful to preserve the minor details and generate clean-and-clear images. Similarly, Figure 4 shows the deraining results on Rain100L [20] dataset which consists of comparatively low rain-density images than that of Rain100H. Zooming in, we can see the better quality of derained images with our proposed method than that of the other methods which leave several rain scars over the images.

B. QUANTITATIVE AND QUALITATIVE RESULTS ON REAL-WORLD RAINY IMAGES

In order to verify the applicability of our proposed deraining network on real-life imagery, we use two different datasets provided by [20] and [7].
TABLE 2. Quantitative comparison of generalization performance of different state-of-the-art methods on real-world SPA-Data [7]. Note that the results for first four models are taken from SPANET paper [7].

| Dataset  | Clear [24]  | DDN [19]  | JORDER [20] | DID-MDN [26] | PRENET [9] | Ours   |
|----------|-------------|------------|--------------|---------------|------------|--------|
| SPA-Data | 31.31/0.930 | 33.28/0.941 | 32.16/0.932  | 24.91/0.889   | 33.35/0.933 | 33.46/0.933 |

FIGURE 5. Comparative demonstration of visual quality by different models for a typical real-rainy image provided by SPANET [7]. Compared to the other state-of-the-art methods, DDN [19] and our proposed network has better reconstruction quality. The differences in reconstruction quality can be best viewed when zoomed in.

TABLE 3. Results showing the effect of addition and removal of various modules from the proposed network on high rain (Rain100H) datasets. The metrics are in the form of PSNR/SSIM.

| Dataset        | Front-end only | no multiscale | multiscale in front-end only | multiscale in both(proposed) |
|----------------|----------------|---------------|------------------------------|------------------------------|
| Rain100H       | 30.26/0.910    | 30.82/0.912   | 30.96/0.914                  | 31.07/0.915                  |

1) EVALUATION ON IMAGES PROVIDED BY [20]
This dataset contains real rainy images without any ground truths. So we can only express the visual results. As shown in Figure 6, we compare our derained images with three popular state-of-the-art methods. When zoomed in, we can see the differences and realistic nature of the derained images in terms of removing the rain streaks and background modifications, if any. In the first image, we can see the methods in [19], [20], and our method generate the derained images without any background modifications which could be noticed if compared with the [9] result around the two parallel tree trunks as well as around the grassy area. On the other hand, the rain streaks are better removed by our method than the other three methods. In the second image, one can see that the methods in [19] and [9] are under-deraining leaving the rain streak scars in the derained image whereas [20] -if not over-deraining- and our method generate almost flawless derained images.

2) EVALUATION ON SPA-DATA
This dataset is a large volume of paired real-world rainy image provided by [7]. It contains 1,000 paired test images which we use to quantify the generalization performance of our proposed network and compare the performance with other state-of-the-art methods. In order to compare the generalization capability of different methods, these are trained on their original datasets and tested on spanet test set. Our model is trained on Rain100L dataset and tested on these 1,000 test images. The obtained results in-terms of PSNR and SSIM are expressed in Table 2. Our method performs best in-terms-of PSNR. In-terms-of SSIM, our method achieves slightly lower than that of the best performing method DDN [19]. The qualitative results of these methods on a typical rainy-image is compared with that of our proposed method in Fig 5. In the figure, the longer rain streaks are erased better by DDN [19], however it leaves a lot-of shorter rain streaks compared to others. DID-MDN [26] modifies the texture of derained image and it barely removes the rain streaks. JORDER [20] and PreNet [9] perform competitively and superior to DID-MDN while performing inferior to our method.

V. ABLATION STUDY
In Table 3, we report the results of ablation study showing the importance of several modules in our proposed network. We show the effect of removal and addition of inception-based multi-scale module and refinement module with respect to the front-end module. While removing multi-scale module, we add a convolutional layer with 3 × 3 filter size in order to generate the total number of feature maps desired by the successor modules in the network. We can clearly see the benefit of addition of inception-based
FIGURE 6. Comparative demonstration of visual quality by different models for real-rainy images provided by [20]. Compared to the other state-of-the-art methods, our proposed network has better reconstruction quality either in the form of removing rain drops completely or avoiding the background modifications. Best viewed when zoomed in.

FIGURE 7. Comparative demonstration of visual quality by different parts of our network in Rain100H [20] dataset. The quantitative results expressed in Table 3 can easily be visualized if the figure is zoomed in and noticed inside the red circle. Note that "multi-scale in both" represents our proposed network.

VI. RECURSION DEPTH VS. NETWORK PERFORMANCE
Our proposed network is recursive in nature which allows one to increase the network length without increasing the total number of model parameters. So, the recursion depth required to complete one iteration of training can be taken as a hyper-parameter. We perform a detailed study to assess the performance of our proposed network with respect to the recursion depth. As shown in Table 4, we vary the recursion depth from one to seven and record the corresponding test performances in terms of PSNR and SSIM. We see the network performance increases from one to five and it starts to decrease. Our network surpasses the results reported by other state-of-the-arts methods in Table 1 only with the recursion depth value of two.

It is obvious that the network training and testing time increases with the increase in recursion depth. However, the PSNR/SSIM might be better with increasing its’ value to a certain level. So, it is the user’s flexibility to set the value of recursion depth based on their available computational resources and desired quality of deraining.

VII. NETWORK LENGTH VS. NETWORK PERFORMANCE
In this analysis we investigate the effect of length of our front-end module on the network performance. In our original proposed network, we use five residual blocks and fusion blocks each. As our fusion block is responsible for majority of computational load of the network, we assess the performance in terms of addition and removal of those blocks. While doing so, the connected residual blocks are also added or removed along with the fusion blocks. The recursion depth of the network while performing this study is kept constant to five as our network performs best at this value as shown in Table 4. In Table 5, the quantitative results of this experiment are expressed as length of front-end module versus the performance of the network. In general, we can see the increasing trend of performance in-terms of PSNR/SSIM
TABLE 4. Recursion depth Vs. performance of the proposed network on Rain100H dataset. The metrics are in the form of PSNR/SSIM.

| Depth | PSNR/SSIM |
|-------|-----------|
| 1     | 30.62/0.909 |
| 2     | 30.44/0.908 |
| 3     | 30.78/0.912 |
| 4     | 30.87/0.913 |
| 5     | **31.07/0.915** |
| 6     | 31.04/0.915 |
| 7     | 30.92/0.913 |

TABLE 5. Length of the front-end module in-terms-of the number of fusion modules Vs. the corresponding performance on Rain100H dataset. The metrics are in the form of PSNR/SSIM.

| No. of Fusion modules | PSNR/SSIM |
|-----------------------|-----------|
| 1                     | 30.62/0.909 |
| 2                     | 30.84/0.912 |
| 3                     | 30.85/0.910 |
| 4                     | 30.88/0.913 |
| 5                     | **31.07/0.915** |
| 6                     | 30.92/0.913 |

TABLE 6. Comparing the performance of using multiscale structure in [20] to that of ours (inception-based multiscale). We replace both of our inception-based multiscale module with that of [20] keeping all other network parameters same to show the effectiveness of using inception-based multiscale. Note that the hardware and software used during this experiment were also same for all cases (hardware: Nvidia Quadro GP100 GPU, environment: Pytorch version-0.4).

| Multiscale structure | PSNR/SSIM | Frame rate |
|----------------------|-----------|------------|
| [20]                 | 30.22/0.90 | 3.57       |
| ours                 | 31.07/0.915 | 3.84       |

TABLE 7. Comparison of performances of the light-weight versions with that of the originally proposed network. The quality metrics are in the form of PSNR/SSIM. The Frame rate is the number of images derained per-unit time (frames/second). Note that Original, LW1, and LW2 represent our originally proposed network, Light-Weight version-1, and Light-Weight version-2 respectively. Note that the hardware and software used during this experiment were same for all cases (hardware: Nvidia Quadro GP100 GPU, environment: Pytorch version-0.4).

| Network | PSNR/SSIM | LW1   | LW2   | Frame Rate |
|---------|-----------|-------|-------|------------|
| Original | 31.07/0.915 | 30.50/0.907 | 30.35/0.908 | 3.54 |
| LW1     | 30.62/0.909 | 30.84/0.912 | 30.85/0.910 | 3.57 |
| LW2     | 30.88/0.913 | 3.54 |

IX. CONCLUSION AND FUTURE WORK
In this work, we proposed a new and flexible deraining network based on sub-pixel convolutions and context aggregation, which consists of two different modules called front-end module and refinement module. Our front-end module generates the deraining results which are then refined by the refinement module. The cascaded connection of these two modules are trained recursively where we can set the recursion depth as a hyper-parameter based on the availability of computational resources and desired level of accuracy. Extensive experiments on several benchmark dataset along with comprehensive ablation study proves that the proposed network works very well for deraining and outperforms other state-of-the-art methods. Currently, ample amount of research work is being focused on supervised deraining approach. However, there is very little concern and work towards the semi-supervised and un-supervised methods for single image deraining. Such methods hold very big importance as the rainy-clear image pairs for the real-world scenario is hard to prepare. In future, we will extend our single image deraining work from supervised to semi-supervised framework such that the deraining method would transfer the knowledge learned from paired synthetic rainy-clear images to unpaired real-world rainy images.

REFERENCES
[1] Z. Shen, W. Wang, X. Lu, J. Shen, H. Ling, T. Xu, and L. Shao, “Human-aware motion deblurring,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 5572–5581.
[2] P. Liu, J. Janai, M. Pollefeys, T. Sattler, and A. Geiger, “Self-supervised linear motion deblurring,” IEEE Robot. Autom. Lett., vol. 5, no. 2, pp. 2475–2482, Apr. 2020.
[3] H. Zhang and V. M. Patel, “Densely connected pyramid dehazing network,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 3194–3203.
[4] K. Katiyar and N. Verma, “Single image haze removal algorithm using color attenuation prior and multi-scale fusion,” Int. J. Comput. A ppl., vol. 141, no. 10, pp. 37–42, May 2016.
[5] M. I. Anwar and A. Khosla, “Vision enhancement through single image fog removal,” Eng. Sci. Technol., Int. J., vol. 20, no. 3, pp. 1075–1083, Jun. 2017.
[6] R. Qian, R. T. Tan, W. Yang, J. Su, and J. Liu, “Attentive generative adversarial network for raindrop removal from a single image,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 2482–2491.
[7] 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.
[8] S. Li, X. Cao, I. B. Araujo, W. Ren, Z. Wang, E. K. Tokuda, R. H. Junior, R. Cesar-Junior, J. Zhang, and X. Guo, “Single image deraining: A comprehensive benchmark analysis,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 3838–3847.

[9] D. Ren, W. Zhao, Q. Hu, P. Zhu, and D. Meng, “Progressive image deraining networks: Toward a simpler baseline,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 3937–3946.

[10] R. Hintze and B. Morse, “Shadow patching: Guided image completion for shadow removal,” in Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), Jan. 2019, pp. 1999–2008.

[11] L. Qi, J. Tian, S. Huang, X. R. Ding, and W. H. L. Lau, “DeshadowNet: A multi-context embedding deep network for shadow removal,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 4067–4075.

[12] P. C. Barmun, S. Narasimhan, and T. Kanade, “Analysis of rain and snow in frequency space,” Int. J. Comput. Vis., vol. 86, nos. 2–3, p. 256, 2010.

[13] J. Bossu, N. Hautière, 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.

[14] S. You, R. T. Tan, R. Kawakami, Y. Mukiagawa, and K. Ikeuchi, “Adherent raindrop modeling, detection, and removal in video,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 9, pp. 1721–1733, Sep. 2016.

[15] V. Santhaseelan and V. K. Asari, “Utilizing local phase information to remove rain from images,” Int. J. Comput. Vis., vol. 112, no. 1, pp. 71–89, Mar. 2015.

[16] W. Yang, R. T. Tan, S. Wang, Y. Fang, and J. Liu, “Single image deraining: From model-based to data-driven and beyond,” arXiv:1912.07150, [Online]. Available: http://arxiv.org/abs/1912.07150.

[17] Y. Luo, Y. Xu, and H. Ji, “Removing rain from a single image via discriminative space-time filtering,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 3397–3405.

[18] 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.

[19] 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.

[20] 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 Int. Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 1357–1366.

[21] G. Li, X. He, W. Zhang, H. Chang, L. Dong, and L. Lin, “Non-locally enhanced encoder-decoder network for single image de-raining,” in Proc. ACM Multimedia Conf. Multimedia Conf. MM, 2018, pp. 1056–1064.

[22] X. Li, J. Xu, Z. Lin, H. Liu, and H. Zha, “Recurrent squeeze-and-excitation context aggregation net for single image deraining,” in Proc. Eur. Conf. Comput. Vis. (ECCV), pp. 254–269, 2018.

[23] Y. Wang, J. Liu, S. Wang, and Z. Guo, “Scale-free single image deraining via capability-enhanced recurrent wavelet network,” IEEE Trans. Image Process., vol. 28, no. 6, pp. 2948–2961, Jun. 2019.

[24] X. Fu, J. Huang, X. Ding, Y. Liao, and J. Paisley, “Clearing the skies: A deep network architecture for single-image rain removal,” IEEE Trans. Image Process., vol. 26, no. 6, pp. 2944–2956, Jun. 2017.

[25] 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.

[26] 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.

[27] X. Wang, R. Girshick, A. Gupta, and K. He, “Non-local neural networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 7794–7803.

[28] 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.

[29] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 4700–4708.

[30] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 1874–1883.

[31] A. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi, “Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resolution convolution and convolution resize,” 2017, arXiv:1707.02937. [Online]. Available: http://arxiv.org/abs/1707.02937.