Memory-guided Image De-raining Using Time-Lapse Data

Jaehoon Cho, *Member, IEEE,* Seungryong Kim, *Member, IEEE,* and Kwanghoon Sohn, *Senior Member, IEEE*

***Abstract*—This paper addresses the problem of single image de-raining, that is, the task of recovering clean and rain-free background scenes from a single image obscured by a rainy artifact. Although recent advances adopt real-world time-lapse data to overcome the need for paired rain-clean images, they are limited to fully exploit the time-lapse data. The main cause is that, in terms of network architectures, they could not capture long-term rain streak information in the time-lapse data during training owing to the lack of memory components. To address this problem, we propose a novel network architecture combining the time-lapse data and, the memory network that explicitly helps to capture long-term rain streak information. Our network comprises the encoder-decoder networks and a memory network. The features extracted from the encoder are read and updated in the memory network that contains several memory items to store rain streak-aware feature representations. With the read/update operation, the memory network retrieves relevant memory items in terms of the queries, enabling the memory items to represent the various rain streaks included in the time-lapse data. To boost the discriminative power of memory features, we also present a novel background selective whitening (BSW) loss for capturing only rain streak information in the memory network by erasing the background information. Experimental results on standard benchmarks demonstrate the effectiveness and superiority of our approach.**

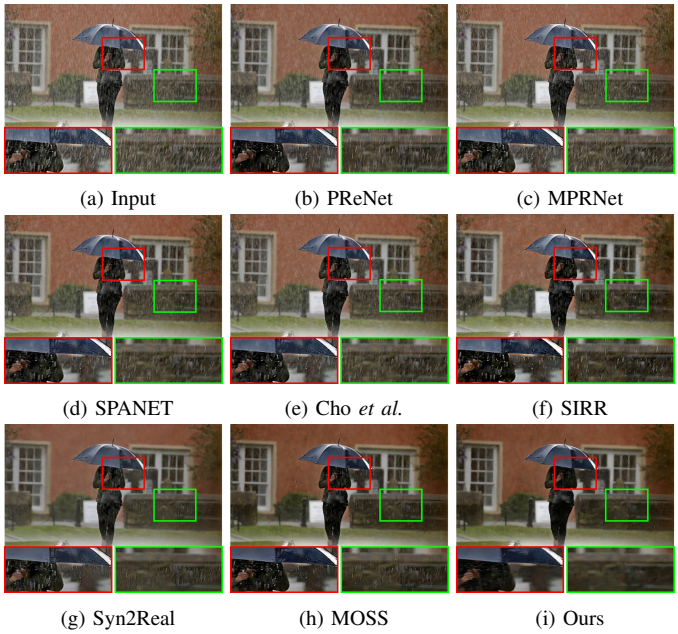
***Index Terms*—Convolutional neural networks (CNNs), image de-raining, memory network, time-lapse data**

I. Introduction

FOR images captured in rainy environments, the performance of numerous computer vision and image processing algorithms, such as object detection [1]–[3], visual tracking [4], or semantic segmentation [5], [6], is often significantly degraded. Image de-raining, aiming to restore a clean (or de-rained) image from the rain image, has thus attracted much attention from researchers in computer vision and image processing community as an essential pre-processing step.

With the significant success of deep convolutional neural networks (CNNs), many attempts been made to solve the image de-raining problem using deep CNNs [6]–[19]. Owing to the powerful feature representation of CNNs, these studies showed a significant performance improvement compared to conventional handcrafted methods [20]–[24]. Most existing methods using CNNs relied on synthetic rain images to train their networks because it is challenging to obtain paired real rain images and corresponding clean images. As easily expected, they exhibit limited performance when handling real rain images because the synthetic data cannot fully reflect various realistic rain streaks such as rain shape, direction, and intensity [6], [10], [12], [26].

To address this problem, several studies [25]–[27] have been developed to learn the rain streak priors with both synthesized paired data and unpaired real data. However, because they still rely on many synthetic rain images, they tend to fail when dealing with real rain images that have never been encountered during training [30]. The alternative direction is to utilize real-world *time-lapse* data with constant background, except for time-varying rain streaks, because the consistent background information can be relatively easily modeled with the data. SPANET [12] constructed a large-scale dataset of real rain/clean image pairs using time-lapse data. Cho *et al.* [6] proposed a background consistency loss to estimate the



(a) Input (b) PReNet (c) MPRNet

(d) SPANET (e) Cho *et al.* (f) SIRR

(g) Syn2Real (h) MOSS (i) Ours

Fig. 1. **Visual results of de-raining methods on real rain image degraded with various types of rain streaks.** (a) Input image, and results of (b) PReNet [16], (c) MPRNet [19], (d) SPANET [12], (e) Cho *et al.* [6], (f) SIRR [25], (g) Syn2Real [26], (h) MOSS [27], and (i) Ours. Contrary to the existing methods, the proposed method fully exploits the long-term rain streak information in the time-lapse data during training owing to the memory networks, thereby showing better performance.

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (NRF2021R1A2C2006703). The work of Seungryong Kim was supported by the MSIT, Korea, under the ICT Creative Consilience program (IITP-2022-2020-0-01819) supervised by the IITP.

Jaehoon Cho and Kwanghoon Sohn are with the School of Electrical and Electronic Engineering, Yonsei University, Seoul 120-749, South Korea (e-mail: {rehoon,khsohn}@yonsei.ac.kr).

Seungryong Kim is with the Department of Computer Science and Engineering, Korea University, Seoul 02841, Korea. (E-mail: seungryong_kim@korea.ac.kr).

Corresponding author:Kwanghoon Sohn.

consistent backgrounds of input images sampled from time-lapse data. They achieved improved performance for real-world rain images using real-world data. Although they proved that using the time-lapse data for training enables overcoming the limitations of using synthetic data, their architectures are limited to fully utilizing the time-lapse data, because they rarely consider the long-term rain streak information across the time-lapse data.

In this paper, we propose a novel network architecture and framework for single image de-raining based on a combination of the time-lapse data and, a memory network that fully exploits the long-term rain streak information. The key insight of our work is to store rain streak-relevant features in the memory network by removing consistent background information of time-lapse data. Specifically, our framework consists of the encoder–decoder networks and a memory network. The extracted features from the encoder, namely queries, are input into the memory network. Memory network, containing several memory items, stores and updates rain streak feature representations. This network explicitly retrieves relevant memory items with respect to queries, and thus memory items can represent various rain streaks included in the time-lapse data.

In addition, to capture only rain streaks-relevant features in the memory network, we propose a novel background selective whitening (BSW) to whiten the background information included in the queries so that the rain streak information is only stored in the memory network. These whitened queries allow the storage of diverse rain streaks such as direction, shape, density, and scale for different rain streaks into memory items without ground-truth paired data during training (update) and accessing them at test time (read). By incorporating our memory network with the BSW loss, the proposed method can effectively remove rain streaks and generate clearer background information.

Experimental results on several standard benchmarks, including synthetic datasets [8]–[10] and real datasets [12], show the improvement of the proposed memory network and demonstrate the improved generalization ability on real data.

Our main contributions are highlighted as follows.

- We propose a novel network architecture combining the time-lapse data and, the memory network that explicitly helps to capture long-term rain streak information.
- To encourage the memory network to capture only rain streak-relevant features, we introduce a novel BSW loss that removes the consistent background information in the time-lapse data.
- We conducted extensive experiments on various datasets to demonstrate that the proposed approach outperforms recent state-of-the-art methods both quantitatively and qualitatively.

The remainder of this paper is organized as follows. Section II describes the related works. The proposed method is presented in Section III. Extensive performance validation is provided in Section IV, including an ablation study and comparison with the state of the art. Finally, Section V concludes this paper.

II. Related Work

A. Single Image De-raining

In general, traditional single-image de-raining methods have been designed to explicitly model the physical characteristics of rain streaks. For instance, they adopted sparse coding [1], dictionary learning [20], Gaussian mixture model [21], and low-rank constraints [22]. Such methods often fail under complex rain conditions, and show over-smoothed images.

By leveraging convolutional neural networks (CNNs), recent learning-based methods have been proposed in numerous studies. Fu *et al.* [8] first proposed de-raining network that decomposed the image into high- and low-frequency components and processed the high-frequency components using CNN. Yang *et al.* [11] estimated binary rain masks and rain, using a contextualized dilated network and they extended a prior work through a recurrent structure [10]. Zhang and Patel [9] presented a density-aware multi-stream de-raining network that uses a density cue from a rain density classifier. A recurrent framework leveraging a squeeze-and-excitation network [45], a progressive network [16], and a wavelet transform [17] were introduced to gradually remove rain streaks. Wang *et al.* [18] proposed a rain convolutional dictionary network in which the rain shapes were encoded. Yang *et al.* [13] designed fractal band learning network based on frequency band recovery. Hu *et al.* [46], [47] proposed a depth-guided attention mechanism to remove rain and fog. Lin *et al.* [48] designed a sequential dual attention deep network to capture the distribution of rain streaks within a rainy image. Wang *et al.* [49] proposed a two-branch encoder design for capturing the contextual regions by learning the soft attention mask. Yasarla *et al.* [63] proposed an image quality-based method that learns the quality or distortion level of each patch in the rain image. MPRNet [19] proposed a multi-stage architecture that progressively learns restoration functions for the degraded inputs. Several studies have used synthetic datasets and un-labeled real-world images to adapt real diverse rain streaks by designing an expectation maximization algorithm [25] and the Gaussian process [26]. More reviews of image de-raining methods are summarized well in [30], [50].

Aforementioned approaches rely on large amounts of synthetic rain images for training, limiting their ability to handle real rain images. Unlike the previous studies, our method uses only real-world time-lapse data for training.

B. Video De-raining

Early methods, such as those proposed by Garg and Nayar [31]–[33], studied the visual effects of rain drops and developed a rain detection method using a physics-based motion–blur model. In addition, various video de-raining methods have been proposed to incorporate the spatial and temporal properties of rain streaks using k-means clustering [34], statistical frequencies [35], low-rank hypothesis [36], [38], tensor-based [37], GMM [39], and sparse coding [40]. Recently, the CNN based methods have been investigated. Chen *et al.* [41] proposed a framework using superpixel segmentation to handle torrential rain with opaque rain occlusion. Liu *et al.* [42] designed a joint recurrent rain removal

and reconstruction network that incorporates spatial texture appearances and temporal coherence. Liu *et al.* [43] developed a DRR network to handle dynamic video contexts. Yang *et al.* [44] presented a two-stage recurrent network with dual-level flow regularization. Although they can leverage the temporal information by analyzing the difference between adjacent frames, these methods cannot be directly applied to single-image de-raining because of the lack of temporal knowledge.

C. Using Time-lapse Data

Various CNN-based methods adopt the time-lapse data to exploit the structure preservation properties in which background information is constant while temporal context changes over time, such as time-lapse video generation [64]–[66] and intrinsic decomposition [67]–[69]. Similar to our proposed method, several efforts have been dedicated to improving single-image de-raining [6], [12]. SPANET [12] constructed a large-scale real-world rain/clean paired dataset using time-lapse data and proposed a spatial attentive network (SPANet) that removed rain streaks in a local-to-global spatial attentive manner. Cho *et al.* [6] introduced a large-scale time-lapse dataset and exploited the dataset for a estimating the consistent background without ground truth. They [6], [12] showed the benefits of exploiting the time-lapse data. However, their architectures were limited in their ability fully to exploit time-lapse data because they rarely consider the long-term rain streak information across time-lapse data. In contrast, our proposed method involves a novel network architecture based on memory networks, enabling the exploitation of the long-term rain streak information.

D. Memory Networks

The memory network [51], [52] is a trainable module that stores information in external memory and reads the relevant content from the memory. Weston *et al.* [51] first introduced a memory network. Graves *et al.* [53] introduced the application of external memory to extend the capability of neural networks. To record information more stably, Santoro *et al.* [54] proposed a memory-augmented neural network to rapidly update new data for a one-shot learning problem. Owing to its flexibility, it has been widely adopted for solving various vision problems including movie understanding [55], video object segmentation [56]–[58], image generation [59], VQA [60], and anomaly detection [61], [62].

Very recently, MOSS [27] was proposed as a de-raining method using memory networks. MOSS [27] is conceptually similar to our method in that it also adopts the memory network. However, our method differs from MOSS due to the following reasons. (i) While MOSS [27] mainly focused on storing the rain streaks on the memory network in a supervised manner, we proposed a combination of the memory network and the time-lapse data, enabling us to train in an unsupervised manner. (ii) MOSS [27] often fails to estimate rain streaks because memory features are mixed with the background. In contrast, our method more effectively estimates the rain streaks thanks to the newly proposed background whitening selective loss that removes the background information in the memory features. (iii) In terms of the network architecture, our method is designed with a Siamese structure that can utilize the consistency between input images, enabling learning without ground truth. (iv) The proposed method achieves more appealing results and shows the improved generalization ability on both synthetic data and real-world data.

III. Proposed Method

A. Motivation and Overview

The single image de-raining task aims to remove the rainy effect and recover a rain-free background in a rainy image. An image degraded by a rainy artifact is generally formulated as

$$I = B + R, \tag{1}$$

where B and R are the background or de-rained image and rain streak image, respectively. Decomposing the image I into B and R is notoriously challenging, because it is a highly ill-posed problem [10], [16], [30].

To solve this problem, existing CNN-based methods [7]–[10], [12], [14]–[16], [18], [25], [26] attempted to estimate a mapping function from I to B (or R). Most of the methods are formulated to only use synthetic rain/clean pairs as supervised signals, and they thus often fail to deal with real rain images that have never been encountered during training [30]. To address this issue, several studies [6], [25]–[27] used both synthesized paired data and unpaired real data. Furthermore, SPANET [12] and Cho *et al.* [6] exploited real-world time-lapse data for training. Although they apparently showed improved generalization ability on real data, their network architecture was limited to fully exploiting rain streak information shown in the time-lapse data during training because of the lack of long-term memory components.

To overcome this issue, we propose a novel network architecture combining the time-lapse data and the memory network that fully makes use of long-term rain streak information during training, as shown in Fig. 2. Specifically, our framework consists of the encoder–decoder networks and memory network. Our memory network contains several memory items, each of which stores rain steak-aware feature representations. The features from the encoders, namely queries, are used to read and update the rain steak-aware features in the memory. The decoders then take them as inputs to estimate the final rain streaks. This memory network helps to *explicitly* consider the long-term rain streak information across the time-lapse data, thereby improving time-lapse data utilization for training. For the memory network to only contain rain-streak-relevant features while erasing the background information in queries, which improves the discriminative power of memory item features, we further present a novel background selective whitening (BSW) loss function, inspired by [70]. The proposed loss disentangles the covariance extracted from the queries into the encoded background and rain streaks information, and then selectively suppresses only the background covariance.

Specifically, as training data, we use a set of time-lapse data $\mathbf{D} = \{D_n\}_{n=1,\dots,N}$ and $D_n = \{I_{t,n}\}_{t=1,\dots,T}$, where N is the

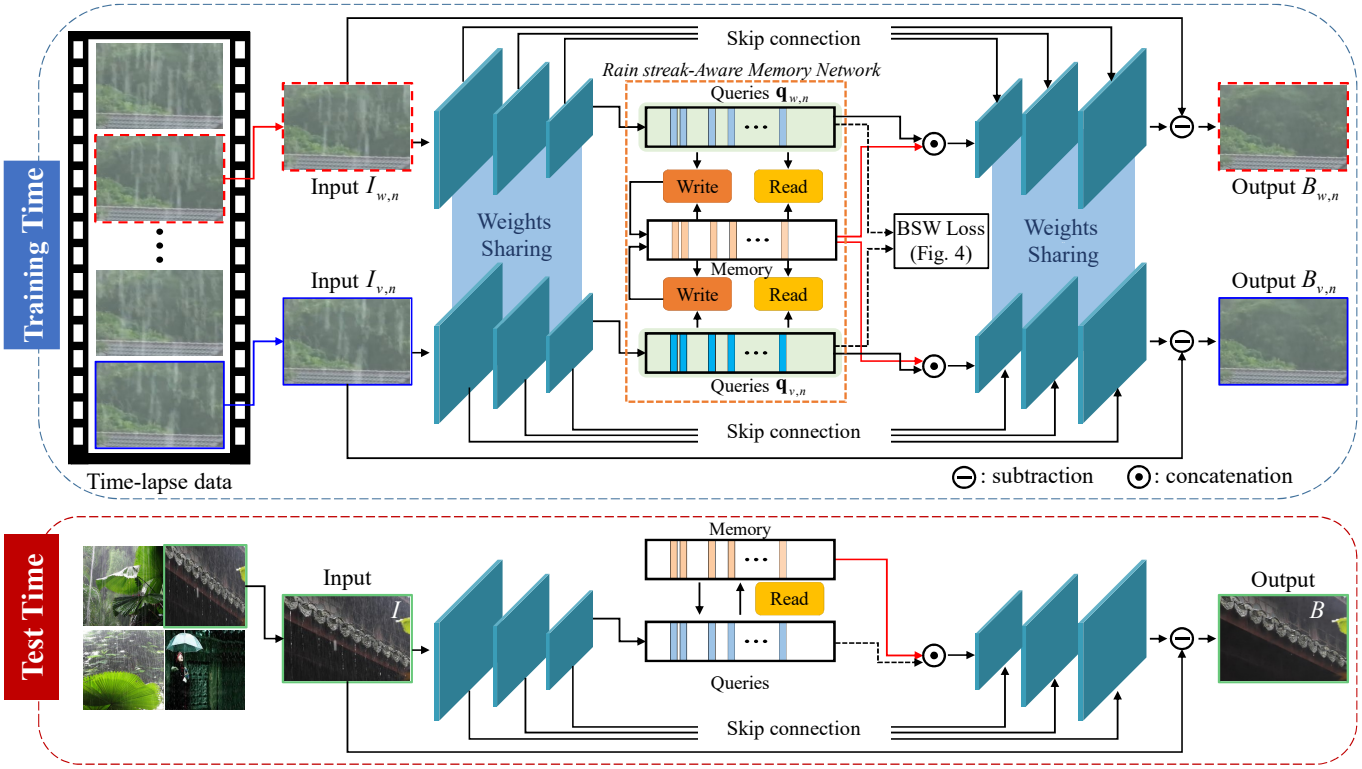


Fig. 2. **Overall network architecture.** The encoder extracts the feature as a query $\mathbf{q}$. The memory network performs read and update operations for reading and storing prototypical rain streak features and covers long-term information. The retrieved memory items from the memory network and query are concatenated and then fed into a decoder to estimate the rain streaks. The de-rained (clean) image $\hat{B}$ is obtained by subtracting the decoder output R from the input rain image I.

number of all time-lapse data, T is the total time of a time-lapse data, n denotes the index of scenes, and t denotes the time. We denote by $I_{t,n}$ and $\mathbf{q}_{t,n}$ an input rain image and a corresponding feature, respectively, at time t in the n^{th} scene. Our objective is to infer a de-rained image $B_{t,n}$ for each $I_{t,n}$ through the proposed method.

B. Network Architecture

In this section, we begin with a description of the proposed network architecture. We largely follow the encoder–decoder network, which has been widely adopted for existing single image de-raining [6], [9], [63]. We describe a detailed description of our network architecture in Table I. Unlike the generative adversarial networks (GANs) based method [7] requiring discriminator to produce results, our method does not require any discriminator for training. Specifically, for all convolutional layers, the kernel size was set to 3×3. In the encoder, all max-pooling layers have a kernel size and stride set to 2×2 and 2, resulting in the output features being down-scale by a factor of 2. The encoder inputs a rain image $I_{t,n}$ and then extracts the feature $\mathbf{q}_{t,n} \in \mathbb{R}^{H \times W \times C}$, which can be used as a query for the memory network, where H, W, and C are the height, width, and number of channels, respectively. We denote by $\mathbf{q}^k_{t,n} \in \mathbb{R}^C$ for $k = 1, ..., K$, where $K = H \times W$, each query of size $1 \times 1 \times C$ in the query map. The query is then inputted to the memory network to read or update the memory items, such that it records prototypical rain streak information. Note that, for simplicity, the subscripts t and n are omitted because the proposed network repeats the same process for each t and n in the memory network. Detailed descriptions of the memory network are presented in the following sections. In the decoder, each layer is composed of 3×3 deconvolution and convolution layers followed by ReLU, which is connected to the encoder using skip connections. The deconvolution layer, implemented with transposed convolutional layers, has an upscaling factor of 2. The decoder inputs the retrieved memory items from the memory network and encoded features $\mathbf{q}$ to produce rain $\hat{R}$. Finally, the de-rained image $\hat{B}$ is obtained by subtracting $\hat{R}$ from input rain image I. Next, we describe how the memory network captures the rain streak information through the read and update process.

C. Rain Streak-Aware Memory Network

In our encoder-decoder networks, we design a memory network containing M memory items to record various prototypical rain streak information and contain long-term rain streak information in time-lapse data. The memory network allows the storage and reading of diverse shapes such as scales, densities, and directions for different rain streaks into memory items during training, while accessing them at inference time. We denote by $\mathbf{p}_m \in \mathbb{R}^C$ for $m = 1, ..., M$ the item in a memory network.

1) Read: To read the appropriate rain streak information in an input rain image, we first compute the similarity between

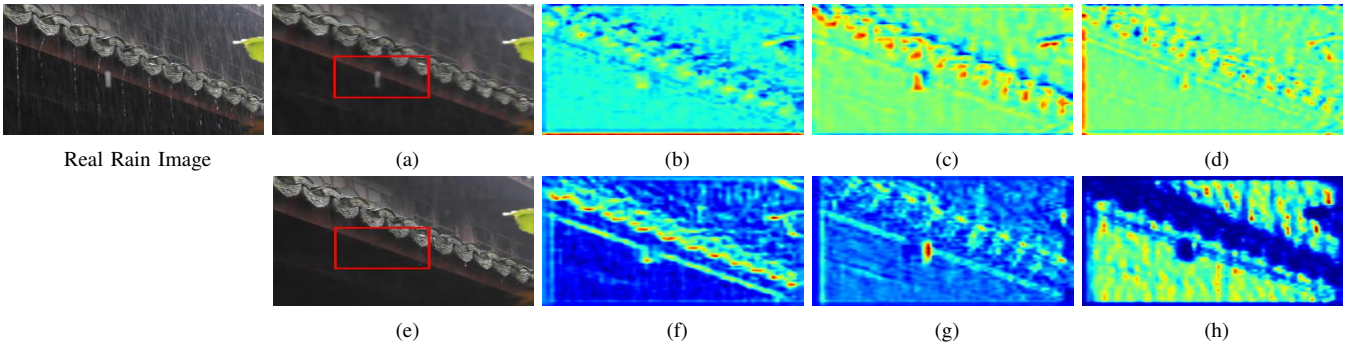


Fig. 3. **Visualization of de-rained results and memory feature maps trained with/without BSW loss.** (a) De-rained result trained without the BSW loss, (b)–(d) memory feature maps trained without BSW loss, (e) de-rained result trained with the BSW loss, and (f)–(h) memory feature maps trained with BSW loss. The proposed loss selectively removes the background information so that only rain streaks are stored in the memory network. Note that the memory feature maps are up–sampled to match the image size (best viewed in color).

TABLE I
THE DETAILS OF ENCODER–DECODER NETWORK: C_IN AND C_OUT DENOTE THE NUMBER OF CHANNELS OF THE INPUT AND OUTPUT FEATURES, RESPECTIVELY. NOTE THAT $\hat{\mathbf{p}}$ IS THE RETRIEVED MEMORY ITEMS. $\{\cdot,\cdot\}$ DENOTES THE CONCATENATION OPERATOR.

Encoder			
Layer	C_in	C_out	Input
conv_E1a	3	64	I
conv_E1b	64	64	conv_E1a
pool_E1	64	64	conv_E1b
conv_E2a	64	64	pool_E1
conv_E2b	64	64	conv_E2a
pool_E2	64	64	conv_E2b
conv_E3a	64	64	pool_E2
conv_E3b	64	64	conv_E3a
pool_E3	64	64	conv_E3b
conv_E4a	64	64	pool_E3
conv_E4b	64	64	conv_E4a
Decoder			
Layer	C_in	C_out	Input
conv_D4a	128	64	{conv_E4b, $\hat{\mathbf{p}}$}
conv_D4b	64	64	conv_D4a
upconv_D3	64	64	conv_D4b
conv_D3a	128	64	{upconv_D3, conv_E3b}
conv_D3b	64	64	Conv_D3a
upconv_D2	64	64	Conv_D3b
conv_D2a	128	64	{upconv_D2, conv_E2b}
conv_D2b	64	64	conv_D2a
upconv_D1	64	64	conv_D2b
conv_D1a	128	64	{upconv_D1, conv_E1b}
conv_D1b	64	64	conv_D1a
Output	64	3	conv_D1b

query $\mathbf{q}^k$ and all memory items $\mathbf{p}_m$, resulting in a read weight matrix $\alpha^{k,m} \in \mathbb{R}^{M\times K}$ as follows:

$$\alpha^{k,m} = \frac{\exp(d(\mathbf{p}_m, \mathbf{q}^k))}{\sum_{m'=1}^{M}\exp(d(\mathbf{p}_{m'}, \mathbf{q}^k))}, \tag{2}$$

where $d(\cdot,\cdot)$ is defined as a cosine similarity:

$$d(\mathbf{p}_m, \mathbf{q}^k) = \frac{\mathbf{p}_m^{\mathsf{T}}\mathbf{q}^k}{\|\mathbf{p}_m\|\,\|\mathbf{q}^k\|}, \tag{3}$$

where T represents the transpose operator. For each query $\mathbf{q}^k$, we read the memory items by taking a weighted average of the memory items $\mathbf{p}_m$ with the corresponding weights $\alpha^{k,m}$, and obtain the retrieved features $\hat{\mathbf{p}}^k \in \mathbb{R}^C$ as follows:

$$\hat{\mathbf{p}}^k = \sum_{m'=1}^{M}\alpha^{k,m'}\mathbf{p}_{m'}. \tag{4}$$

This step is repeated for input rain images sampled by time-lapse data. Through this step, we can fully utilize diverse rain streaks containing all items.

By applying the reading operator to each query, we obtain a transformed feature map $\hat{\mathbf{p}}^k \in \mathbb{R}^{H\times W\times C}$. We concatenate the transformed feature map with the query along the channel dimension and input it to the decoder. This enables the decoder to estimate rain $\hat{R}$ using various rain streak information in the memory items. With the memory network, as shown in Fig. 3 (b)–(d), various types of rain streaks can be captured by each memory items[1].

2) Update: To enhance the memory items during training, we dynamically select and store rain streak-relevant features into the memory network. Similar to the read operation, we compute an updated weight matrix $\beta^{k,m} \in \mathbb{R}^{M\times K}$ between $\mathbf{p}_m$ and $\mathbf{q}^k$:

$$\beta^{k,m} = \frac{\exp(d(\mathbf{p}_m, \mathbf{q}^k))}{\sum_{k'=1}^{K}\exp(d(\mathbf{p}_m, \mathbf{q}^{k'}))}, \tag{5}$$

where we apply the softmax function along the $\mathbf{q}$-direction, as opposed to (2). The updated weight matrix β is used to assign the extracted rain streak-relevant features $\mathbf{q}$ to the relevant memory item. The memory items $\mathbf{p}_m$ are updated using $\mathbf{q}$ weighted by β as follows:

$$\hat{\mathbf{p}}_m = \mathrm{Norm}_{\mathrm{L2}}(\mathbf{p}_m + \sum_{k'=1}^{K}\beta^{k,m}\mathbf{q}^{k'}), \tag{6}$$

where $\mathrm{Norm}_{\mathrm{L2}}$ denotes the L_2 norm. We utilize $\mathbf{q}^{k'}$ to update $\mathbf{p}_m$. We train the memory items with a large number of real-world time-lapse data, enabling the most representative and discriminative rain streak-relevant features to be stored.

During training, we access only a set of real-world time-lapse data without any ground truth to update the memory items assigned to store diverse rain streaks. At inference time, we compute $\mathbf{p}_m$ for all memory items without considering prior information such as rain density levels [9] and a binary map indicating rain streak regions [10], [30], and retrieve the rain streaks using Eq. 2 and Eq. 4. Because our memory network is trained using diverse real-world time-lapse data, this strategy works well on rain images.

[1]Note that feature maps are up–sampled to match the image size.

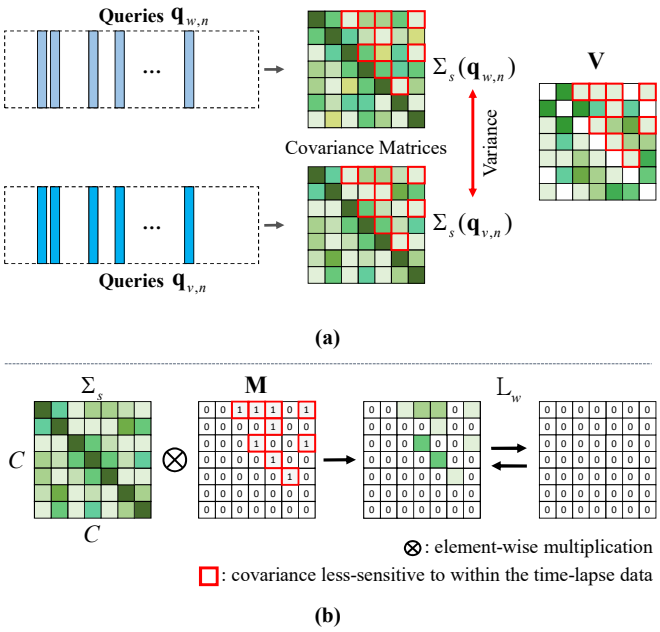


Fig. 4. **Illustration of the background selective whitening (BSW) loss.** (a) The variance matrix $\mathbf{V}$ is computed from the covariance matrices of each queries $(\mathbf{q}_{w,n}, \mathbf{q}_{v,n})$ to identify the same elements (red boxes). Note that these matrices are symmetric. (b) The covariance matrix Σ_s is masked by the matrix $\mathbf{M}$, which belongs to a low variance value, to selectively suppress the same background covariances by the BSW loss $\mathcal{L}_w$.

D. Loss Functions

Following Cho *et al.* [6], we adopt several loss functions to learn the our network architecture, including the memory network. In addition, the proposed method introduces a novel background selective whitening loss to remove the background information between the each queries to improve the discriminative power of the memory items.

1) Background Prediction Loss: Background prediction loss encourages the generation of consistent background images across time-lapse data. It is formulated as the L_1 distance between the estimated background images from time-lapse data, but at different times such that

$$\mathcal{L}_b = \sum_{n\in N} \sum_{\{w,v\}\in T} \sum_{i} \left\| \hat{B}_{w,n}(i) - \hat{B}_{v,n}(i) \right\|_1, \quad (7)$$

where $\hat{B}_{w,n}$ is a background image that is decomposed from $I_{w,n}$. w, v represents the different times in T, and n denotes the indexes of the scene. Here, $\hat{B}_{w,n}(i)$ and $\hat{B}_{v,n}(i)$ are the values at pixel i from the image $\hat{B}_{w,n}$ and $\hat{B}_{v,n}$, respectively.

2) Cross Information Loss: This loss function is designed to encourage the estimated backgrounds to be close to the input images based on the assumption that the overall structure of the estimated backgrounds should be approximated well by input images [68], [69]. This loss helps to understand the structure of the overall layout information. It measures the L_1 distance between the estimated background image and the input image such that

$$\mathcal{L}_c = \sum_{n\in N} \sum_{\{w,v\}\in T} \sum_{i} \left\| I_{w,n}(i) - \hat{B}_{v,n}(i) \right\|_1. \quad (8)$$

Moreover, this loss function allows the network to produce good initial results during the early training phase.

3) Self Consistency Loss: This loss makes the summation of the estimated $\hat{B}$ and $\hat{R}$ to be the input image I such that

$$\mathcal{L}_s = \sum_{n\in N} \sum_{w\in T} \sum_{i} \left\| I_{w,n}(i) - (\hat{B}_{w,n}(i) + \hat{R}_{w,n}(i)) \right\|_1, \quad (9)$$

which acts as a regularizer.

4) Background Selective Whitening Loss: In this section, we introduce a novel additional loss function, called background selective whitening (BSW) loss, to utilize only rain streak information in the memory networks, by erasing the background information in time-lapse data. We use the covariance extracted from the each query to separate them into the encoded background and rain streak. We then propose to handle only the background information, which can be selectively removed, thus improving the de-raining performance. This is possible because each query is from the time-lapse data which contains the consistent background but time varying rain streaks. As shown in Fig. 4, each query is computed to extract each covariance matrices, and the covariance matrices are separated into two groups, including rain streaks and background information, which can be used to selectively whiten (remove) the background information for storing only rain in memory networks. Note that MOSS [27], which uses a memory network for de-raining, often fails to remove rain streaks because the memory features are mixed with background and rain streaks, but the BSW loss efficiently removes the background to improve the discriminative power of the rain steak-relevant features.

Specifically, we extract two covariance matrices by inferring from the queries, extracted from different input rain images $(I_{w,n}, I_{v,n})$, and compute the variance matrix from the differences between two different covariance matrices. We hypothesize that the variance matrix $\mathbf{V}$ implies the sensitivity of the corresponding covariance to the rain streaks. In other words, the covariance elements with high variance values encode different rain streaks, such as density, orientation, and intensity. We define the variance matrix $\mathbf{V} \in \mathbb{R}^{C\times C}$ as $\mathbf{V} = \boldsymbol{\sigma}^2$ from the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$ for each element from two different covariance matrices as follows:

$$\boldsymbol{\mu} = \frac{1}{2}(\boldsymbol{\Sigma}_s(\mathbf{q}_{v,n}) + \boldsymbol{\Sigma}_s(\mathbf{q}_{w,n})), \quad (10)$$

$$\boldsymbol{\sigma}^2 = \frac{1}{2}((\boldsymbol{\Sigma}_s(\mathbf{q}_{v,n}) - \boldsymbol{\mu})^2 + (\boldsymbol{\Sigma}_s(\mathbf{q}_{w,n}) - \boldsymbol{\mu})^2), \quad (11)$$

where $\boldsymbol{\Sigma}_s(\cdot)$ extracts the covariance matrix of the intermediate feature map from each query. As a result, $\mathbf{V}$ consists of elements of the variance of each covariance element across different queries extracted from the input rain images.

We split s into two groups: $G_{low} = \{c_1, ..., c_l\}$ with a low variance value and $G_{high} = \{c_{l+1}, ..., c_h\}$ with a high variance value. We assume that the rain streaks are encoded in the covariance belonging to G_{high}, and consistent background information is encoded in the covariance belonging to G_{low}. Therefore, the background selective whitening loss selectively

suppresses only the background information-encoded covariance. Let the mask matrix $\mathbf{M} \in \mathbb{R}^{C\times C}$ for the BSW loss be such that

$$\mathbf{M} = \begin{cases} 1, & \text{if } \mathbf{V} \in G_{low} \\ 0, & \text{otherwise} \end{cases}. \tag{12}$$

The BSW loss is defined as

$$\mathcal{L}_w = \sum_i \|\mathbf{\Sigma}_{\boldsymbol{s}}(i) \otimes \mathbf{M}(i)\|_1, \tag{13}$$

where $\otimes$ means an element-wise multiplication.

It may be observed that the memory network captured the rain streak in Fig. 3 (b)–(d). This shows that the vanilla memory network could not effectively discriminate the region between the background and rain streaks, which often limits the de-raining models to remove rain streaks. However, with our background selective loss, as shown in Fig. 3 (f)–(h), the network clearly captures the rain streaks in the memory network and enables the effective removal of rain streaks in the real rain image when comparing the highlighted boxes, as shown in Fig. 3 (a) and (e). Note that because the proposed BSW loss depends on the the paired rain images containing the same background but time-varying rain streaks, this loss is well-tailored to time-lapse data.

5) Total Loss: For training our network, the total loss function consists of the aforementioned loss functions. These terms are balanced by the weights λ_b, λ_s, λ_c, and λ_w as

$$\mathcal{L} = \lambda_b\mathcal{L}_b + \lambda_s\mathcal{L}_s + \lambda_c\mathcal{L}_c + \lambda_w\mathcal{L}_w. \tag{14}$$

E. Implementation Details

The proposed networks were trained using the PyTorch [71] library, with an Nvidia RTX TITAN GPU, which requires approximately 24 hours for training. We resized each input image to 256 $\times$ 256 and normalized it to the range [-1, 1]. Because our training data contained various real-world scenes, we do not use data augmentation such as flipping and rotating, as in [6]. We set the height H and width W of the query feature map, number of feature channels C, and memory items M to 32, 32, 32, and 20, respectively. The memory items $\hat{\mathbf{p}}$ are randomly initialized. For the BSW loss, l and h are hyper-parameters, which are empirically set to 2 and 4, respectively. For the loss weighting parameters in Eq. 14, we empirically set λ_b to be 1, λ_s to be 0.1, λ_c to be 0.001, and λ_w. The model is trained using the Adam optimizer [72] with β_1= 0.9 and β_2 = 0.999, with a batch size of 16. We set the initial learning rate to 2e-4 and decayed them using a cosine annealing method [75].

IV. Experiments

A. Experimental Setup

In this section, we demonstrate the results of comprehensive experiments conducted to evaluate the performance of the proposed method, including comparisons with several state of the art single-image de-raining methods that include JCAS [24], DDN [8], DID [9], NLEDN [14], RESCAN [45], PReNet [16], JORDER-E [10], SIRR [25], SPANET [12], Cho *et al.* [6], Hu *et al.* [47], Syn2Real [26], RCDNet [18], MPRNet [19], SSDRNet [48], and MOSS [27]. We used public code or pre-trained models provided by the authors to produce the de-raining results.

In addition, we compared the performance of several video de-raining methods, including Garg *et al.* [33], Kim *et al.* [36], Jiang *et al.* [37], Ren *et al.* [38], Wei *et al.* [39], Li *et al.* [40], and Liu *et al.* [42]. Kim *et al.* [36][2] and Li *et al.* [40][3] provided the rain video dataset containing various forms of moving objects and background scenes, and different types of rain, varying from very light drizzle to heavy rain–storms and vertical rain to nearly horizontal rain.

Using a single trained model, we evaluated both real and synthetic datasets. For quantitative evaluation, we measured the peak signal-to-noise ratio (PSNR) and the structure similarity index measure (SSIM). For evaluating on video dataset, we employed two additional metrics, namely VIF [73] and FSIM [74]. We compared competing methods both qualitatively and quantitatively.

B. Datasets

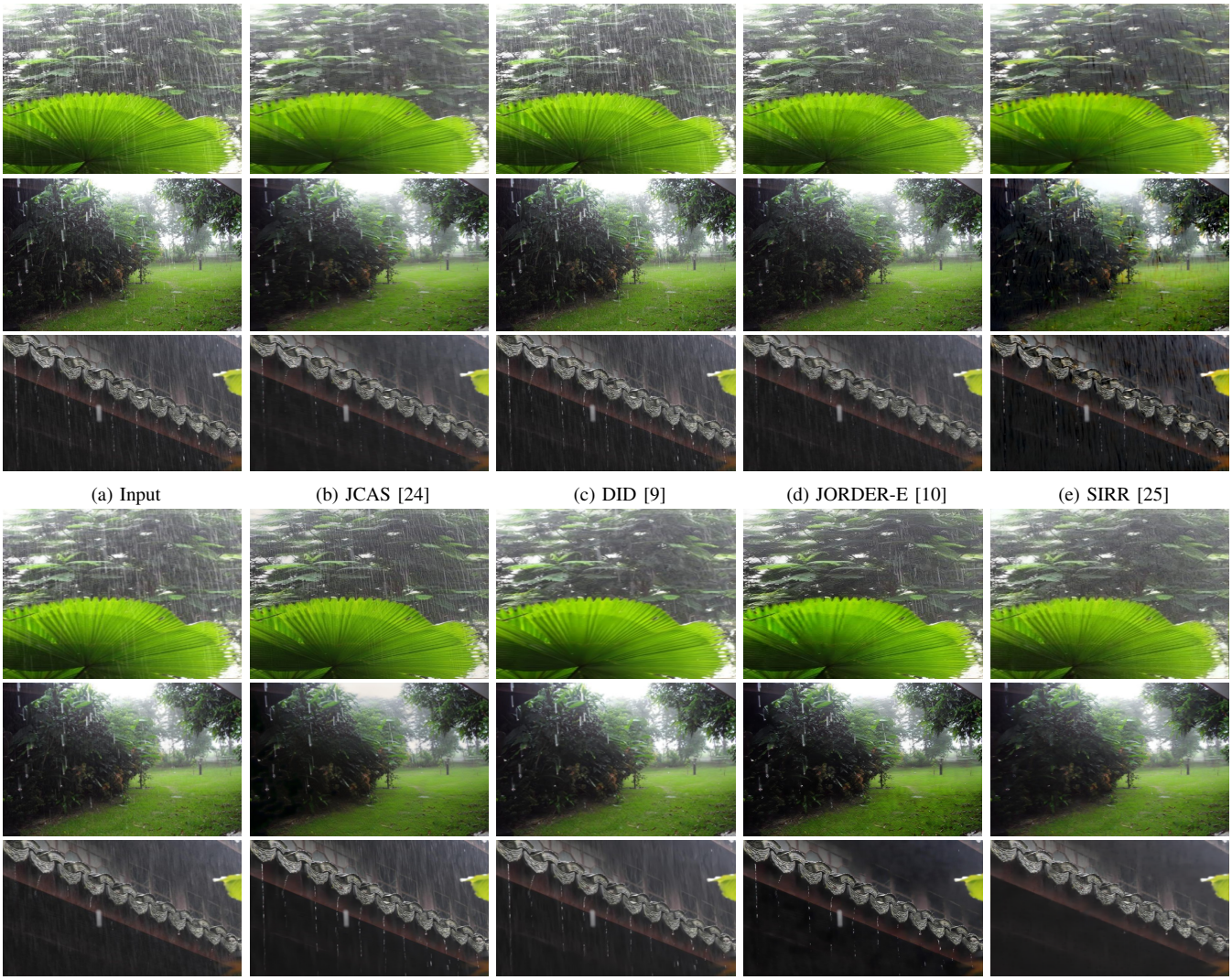
1) Training Dataset: To train our network, we used time-lapse benchmark provided by Cho *et al.* [6]. **TimeLap** provided by Cho *et al.* [6] consists of time-lapse sequences, where rain image pairs comprised 2 images sampled from 30 images from the 186 total scenes. For training and fair comparison, we mainly adopt the same experimental setups as Cho *et al.* [6]. **RealDataset** provided by SPANET [12] consists of 170 real rain videos captured by cell phone or collected from YouTube. **RealDataset** consists of 29,500 rain/clean image pairs image pairs using time-lapse data, which are split into 28,500 for training. Because our method requires time-lapse data without ground truth, we use only time-lapse data for training.

2) Test Dataset: We conducted experiments on real and synthetic datasets, respectively. For the real dataset, we used the **RealDataset** to provide real rain images with realistic ground-truth background images generated by a semi-automatic algorithm [12], thus enabling the quantitative evaluations. This dataset consists of 1,000 pairs for testing at a resolution of 512 $\times$ 512, collected from 170 video sequences. Additionally, we obtained real-world rain images scraped from the Internet and previous studies [9], [10], [12] and used them for qualitative evaluation only. Furthermore, because real rain images often contain *fog*, we also collected real rain images with *fog* from the Internet.

For the synthetic dataset, we used three datasets provided by DDN [8], DID [9], and JORDER-E [10]. **Rain14000** constructed by DDN [8] provides 14,000 rain/clean image pairs synthesized from 1000 clean images with 14 kinds of different rain directions and scales. **Rain12000** constructed by DID [9] provides 12,000 rain/clean image pairs containing rain with different orientations and scales, where the number of images with light, medium, and heavy rain is all 4,000, respectively. As pointed out in [6], [14], because the synthesized examples in Rain100H are inconsistent with real images, we

[2]http://mcl.korea.ac.kr/deraining

[3]https://github.com/MinghanLi/MS–CSC–Rain-Streak–Removal



(a) Input (b) JCAS [24] (c) DID [9] (d) JORDER-E [10] (e) SIRR [25]

(f) RCDNet [18] (g) Syn2Real [26] (h) MPRNet [19] (i) MOSS [27] (j) Ours

Fig. 5. **Qualitative results on real rain images [9], [11].** (a) Input rain images and de-rained results of (b) JCAS [24], (c) DID [9], (d) JORDER [11], (e) SIRR [25], (f) RCDNet [18], (g) Syn2Real [26], (h) MPRNet [19], (i) MOSS [27] and (j) Ours.

used **Rain100**, which consists of 100 light rain images/clean image pairs for training and testing, respectively.

C. Single Image De-raining Results

1) Results on Real-World Data: To evaluate the generalization ability of all competing methods and our method, we first conducted a qualitative evaluation using real rain images collected from the Internet provided by [11], [12], [25]. Fig. 5 shows the de-rained results that our method removes rain streaks well while preserving more detailed information such as background and texture information. In particular, [9], [18] often fail to capture long and thin rain in Fig. 5 (c) and (f). Our result of the second row of Fig. 5 shows the unnatural black streaks on the grass in the right-bottom corner. Recent works including SIRR [25], Syn2Real [26], and MOSS [27] also suffer from these. This phenomenon sometimes arises from the rain formulation Eq. 1 which predicts the rain streak (residual) information and then subtracts it from the input rainy image. However, compared to the state of the art methods, our method effectively handles various types of rain streaks owing to the advantage of being able to store real-world rain streaks with a memory network during training.

Furthermore, we conducted experiments on publically available real-world dataset provided by SPANET [12]. As shown in Fig. 6, traditional hand-crafted method, *that is*, JCAS [24], encountered difficulty in removing rain artifacts. Although CNN–based methods [12], [18] remove rain streaks better than hand-crafted method, they still suffer from the dot-patterned rain streaks. In contrast, our method preserves background details better and effectively removes various rain streaks including dot-patterned and long-shaped rain streaks. In addition, we conducted a quantitative evaluation using the **RealDataset** [12] in the fourth column of Table II. Interestingly, the hand-crafted methods [20], [21], [24] outperformed the CNN–based method [9]. This shows the limitation of the fully supervised learning paradigm because such method tends to fail when dealing with conditions of real rain streaks that have never been encountered during training. Our model achieved the best

(a) Input (b) JCAS [24] (c) JORDER-E [11] (d) PReNet [16] (e) SPANet [12] (f) Cho *et al.* [6]

(g) RCDNet [18] (h) Syn2Real [26] (i) SSDRNet [48] (j) MPRNet [19] (k) MOSS [27] (l) Ours

Fig. 6. **Qualitative results on RealDataset [12].** (a) Input rain images and obtained results of (b) JCAS [24], (c) JORDER-E [10], (d) PReNet [16], (e) SPANet [12], (f) Cho *et al.* [6], (g) RCDNet [18], (h) Syn2Real [26], (i) SSDRNet [48], (j) MPRNet [19], (k) MOSS [27] and (l) Ours.

TABLE II

QUANTITATIVE COMPARISON OF SINGLE IMAGE DE-RAINING USING SYNTHETIC AND REAL-WORLD DATASETS. GT AND T-L DENOTE USING PAIRED GROUND TRUTH DATA AND TIME-LAPSE DATA, RESPECTIVELY. †, ‡, AND ♯ INDICATE THAT THE METHODS REQUIRE ADDITIONAL SUPERVISED CUES SUCH AS, BINARY MASK MAP, RAIN DENSITY LEVEL, AND ATTENTION MAPS, RESPECTIVELY. THE BEST RESULT IS SHOWN IN BOLD, AND THE SECOND–BEST IS UNDERLINED. THE HIGHER THE PSNR AND SSIM IS, THE BETTER.

Benchmark	GT	T-L	Synthetic Dataset						Real-world Dataset	
			Rain14000		**Rain12000**		**Rain100**		**RealDataset**	
			PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
DSC [20]	No	No	27.88	0.839	24.24	0.828	27.16	0.866	34.15	0.927
GMM [21]	No	No	27.78	0.859	25.81	0.834	28.66	0.865	34.30	0.943
JCAS [24]	No	No	26.20	0.847	25.16	0.851	31.42	0.917	34.95	0.945
DDN [8]	Yes	No	28.45	0.889	30.97	0.912	34.68	0.967	36.16	0.946
DID [9]†	Yes	No	26.17	0.887	31.30	0.921	35.40	0.962	28.96	0.941
NLEDN [14]	Yes	No	29.79	0.897	33.16	0.919	36.57	0.975	40.12	0.984
PReNet [16]	Yes	No	32.55	**0.946**	33.17	0.942	37.80	0.981	40.16	0.982
SIRR [25]	Yes	No	28.44	0.889	30.57	0.910	34.75	0.969	35.31	0.941
JORDER-E [10]‡	Yes	No	32.00	0.935	33.98	0.950	38.59	0.983	40.78	0.981
Syn2Real [26]	Yes	No	29.23	0.898	30.90	0.878	36.09	0.967	37.87	0.965
RCDNet [18]	Yes	No	30.66	0.921	31.99	0.921	**40.17**	**0.988**	41.47	0.983
SSDRNet [48]	Yes	No	33.50	0.936	34.32	**0.954**	38.35	0.980	38.07	0.965
MOSS [27]	Yes	No	31.22	0.932	32.87	0.932	37.67	0.974	40.01	0.971
SPANet [12]♯	Yes	Yes	29.85	0.912	33.04	0.949	35.79	0.965	40.24	0.981
Cho et al. [6]	No	Yes	33.73	0.941	33.25	0.935	37.89	0.980	38.54	**0.989**
Ours	No	Yes	**34.02**	0.945	**34.55**	0.949	38.45	0.981	**41.56**	**0.989**

results by leveraging the real-world time-lapse data without ground truth.

2) Results on Synthetic Data: From the first to third row in Table II, we show the quantitative results of recent de-raining methods when trained on synthetic data including **Rain14000**, **Rain100**, and **Rain12000**. We point out that, excluding DID [9] in **Rain14000**, CNN–based methods [6], [8], [10], [12], [14], [16], [18], [25], [26] outperform the hand-crafted methods methods on a synthetic dataset. In addition, the results show that the existing CNN–based methods exhibited significant performance differences depending on the test set. For example, RCDNet [18] has the best performance in the **Rain100** dataset, but the sixth–best performance in **Rain14000**. These results show that supervised learning–based methods vary greatly in performance depending on the characteristics of the synthetic datasets. In contrast, because our method does not rely on supervised learning, it achieved a comparatively consistent performance in various synthetic datasets, showing better generalization ability. Our method achieved the best performance on **Rain14000** and **Rain12000** in terms of PSNR and showed a high generalization capability on other synthetic datasets, including **Rain100**. We believe that models trained with a memory network and BSW loss can alleviate the domain shift of the existing methods.

We further compare the visualization results of our method with those of the state of the art methods in Fig. 7. Our method, designed to focus on real-world rain streaks, can also effectively handle synthetic rain streaks and achieve better results. Our method preserves details while effectively removing rain, demonstrating that our method could discriminate the rain streaks and background scene better than the existing methods.

3) Analysis on Rain Image with Fog: For further analysis, we analyzed whether the proposed method could estimate rain streaks even with fog. We also compared with state of the art methods including Hu *et al.* [47], which proposed a depth-guided attention mechanism to remove rain and fog simultaneously. Fig. 8 illustrated the comparison results. While existing methods are difficult to remove rain streaks with fog,

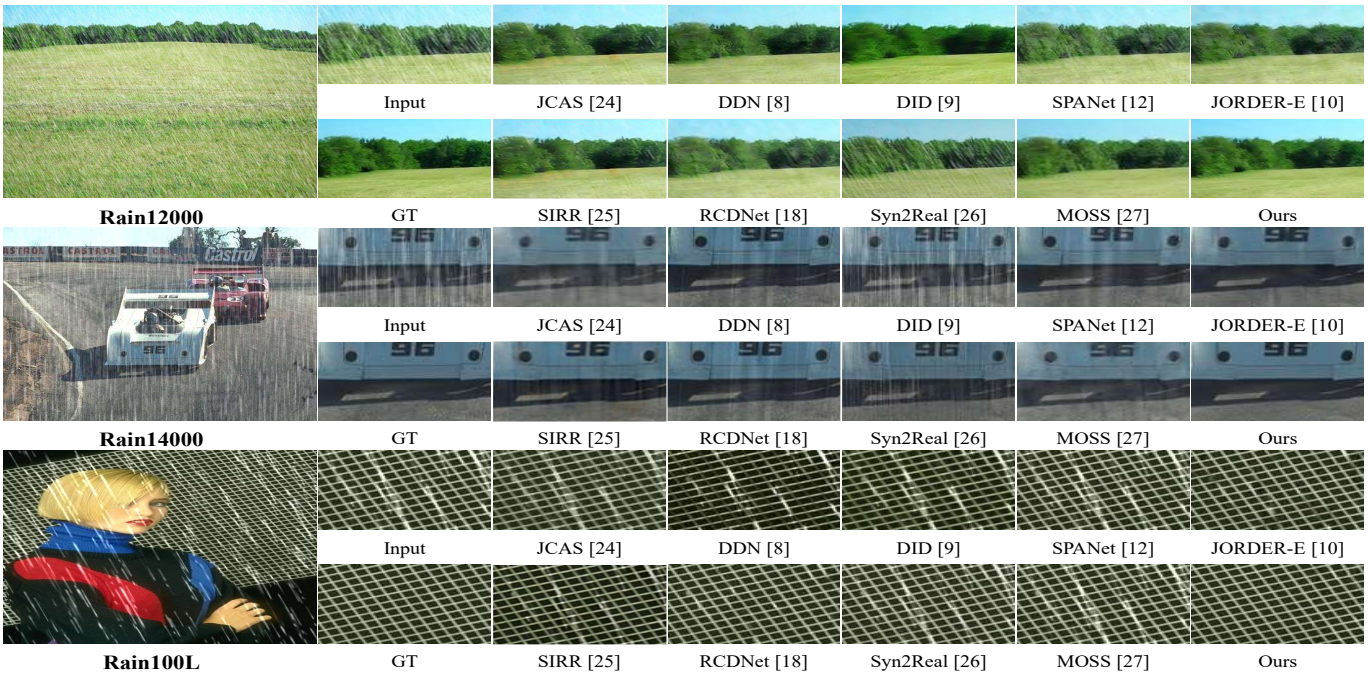


Fig. 7. **Qualitative results on different synthetic datasets including Rain14000, Rain12000, and Rain100.**

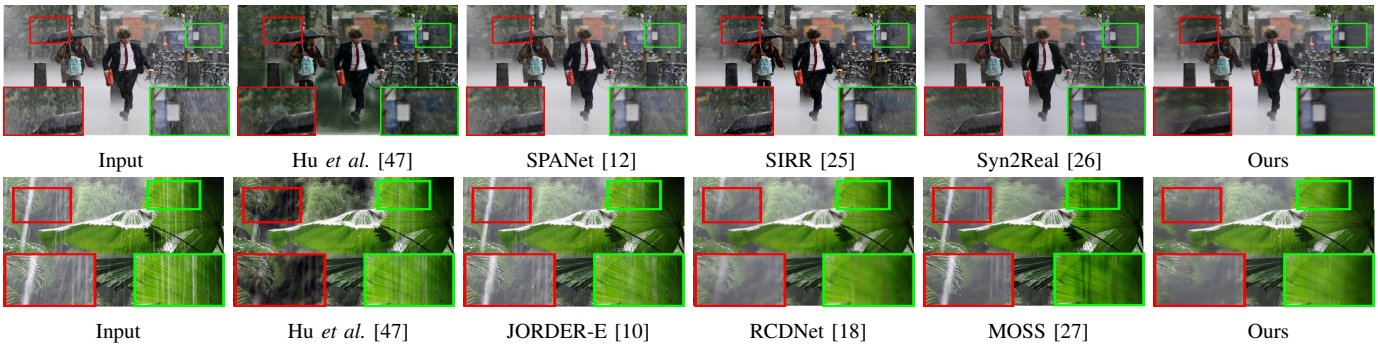


Fig. 8. **Results of the proposed method and the state of the art methods on real rain images accompanied by fog.**

TABLE III
QUANTITATIVE COMPARISON OF VIDEO DE-RAINING METHODS ON A VIDEO DATASET [40].

Metrics	PSNR (↑)	VIF (↑)	FSIM (↑)	SSIM (↑)
SPANet [12]	22.83	0.651	0.953	0.909
Garg *et al.* [33]	24.15	0.611	0.970	0.911
Kim *et al.* [36]	22.39	0.526	0.960	0.886
Jiang *et al.* [37]	24.32	0.713	0.932	0.938
Ren *et al.* [38]	23.52	0.681	0.966	0.927
Wei *et al.* [39]	24.47	0.779	0.966	0.951
Li *et al.* [40]	25.37	0.790	0.980	0.957
Liu *et al.* [42]	22.19	0.555	0.980	0.895
Ours	24.21	0.722	0.967	0.913

our method is successful in removing the rain streaks even with fog. Hu *et al.* [47] tended to leave rain streaks. In addition, they often show the artifacts observed in the object boundaries in the first row in Fig. 8. Alhough Hu *et al.* [47] can handle fog, it is still challenging to remove rain streaks. In contrast, our method can handle rain streaks affected by fog in the real rain images because the proposed memory-based method with BSW loss captured real-world rain streaks well.

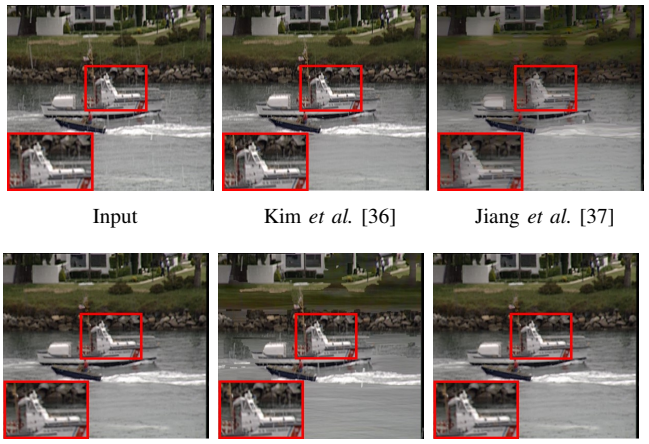


Fig. 9. **Visual comparison on a rain video dataset [40].**

D. Video De-raining Results

Although our method requires a single image at the testing time, it can perform well even for video data. We conducted experiments on videos containing various types of synthetic

TABLE IV
ABLATION STUDY IN TERMS OF THE NETWORK ARCHITECTURE.

	$\mathcal{B}_a$	$\mathcal{B}_b$	$\mathcal{B}_c$	$\mathcal{B}_d$
Encoder-Decoder	✓	✓		
Siamese			✓	✓
Memory		✓		✓
PSNR	36.62	38.03	39.82	41.08
SSIM	0.965	0.972	0.982	0.985

TABLE V
ABLATION STUDY OF LOSS FUNCTIONS.

$\mathcal{L}_s$	$\mathcal{L}_b$	$\mathcal{L}_c$	$\mathcal{L}_w$	PSNR	SSIM
✓	✓			39.92	0.981
✓	✓		✓	40.35	0.982
	✓	✓		40.01	0.982
	✓	✓	✓	40.57	0.984
✓	✓	✓		41.08	0.985
✓	✓	✓	✓	41.56	0.988

rain streaks, as shown in Fig. 9. Kim *et al.* [36], and Li *et al.* [40] did not effectively remove rain. Although the methods, proposed by Jiang *et al.* [37] and Chen *et al.* [41], exhibited a better removing rain streaks, the results showed a blurry artifact in the moving objects. While our method requires only a single image at inference time, our result removes rain effectively and preserves the details of the background and moving object. Table III shows the quantitative comparisons with these competing methods and recent works, on the highway dataset. It should be noted that although all the abovementioned methods require multiple images at training and testing time, our method achieved fairly plausible results using single image at the testing time compared to existing video de-raining methods.

In addition, to compare with the single image de-raining method, we show qualitative results of the video de-raining performance compared to SPANET [12] [4]. We used the pre-trained model provided by SPANET and our pre-trained model, because we want to observe how the single image de-raining peforms well even in video data. Table III shows the quantitative comparisons with SPANET. Thanks to the combination of the proposed memory network and time-lapse data, enabling the exploitation of the long-term rain streak information in time-lapse data, our method outperforms SPANET quantitatively and qualitatively.

E. Ablation Study

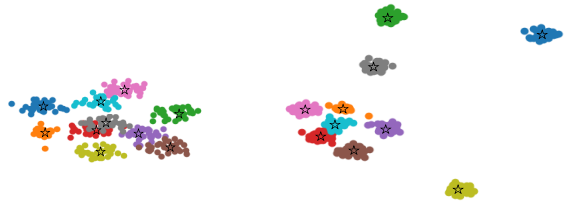
In this section, we conducted an extensive ablation study to verify the importance of each component of the proposed method on the **RealDataset** and real-world rain images.

1) Analysis of Memory Network: To verify the effectiveness of the memory network, we evaluated the different combinations in Table IV. First, $\mathcal{B}_a$ is used as a baseline encoder-decoder without a memory module. $\mathcal{B}_b$ adds a memory network on $\mathcal{B}_a$. These were trained using rain and time-averaged clean images [5] pairs using **TimeLap**. To train the $\mathcal{B}_a$ and $\mathcal{B}_b$, we used the standard $\mathcal{L}_1$ loss to measure the per-pixel reconstruction accuracy [12]. $\mathcal{B}_c$ is a Siamese network based on an encoder–decoder without a memory network. $\mathcal{B}_d$ adds a memory network to $\mathcal{B}_c$. To train $\mathcal{B}_a$ and $\mathcal{B}_b$, we used the loss functions including $\mathcal{L}_b$, $\mathcal{L}_c$, and $\mathcal{L}_s$. The analysis of the proposed BSW loss will be described in the following section. Table IV shows that the proposed Siamese networks exhibited improved the performance compared to a simple encoder–decoder. The model trained with the memory network achieved a substantial accuracy gain over the model without a memory network. Because the memory network enables the learning of various rain streaks, the model with the memory network achieves the best performance, which proves that the memory network is helpful in improving de-raining performance.

2) Analysis of Background Selective Whitening Loss: To verify the effectiveness of the proposed BSW loss, we visualized the distribution of the query features, which were learned

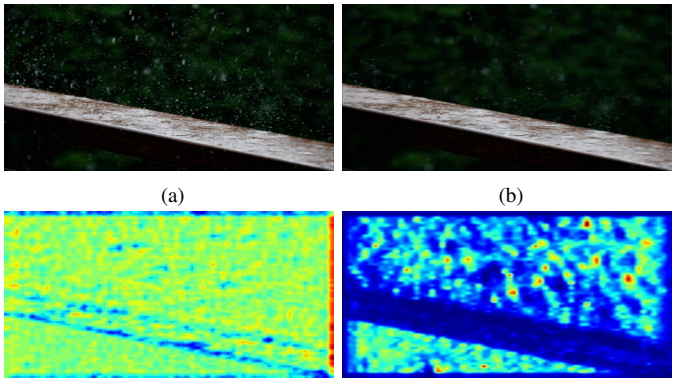
[4] Video result is available at here

[5] Given the characteristics of time-lapse sequences taken from static scenes, simply averaging all frames over time can be used for pseudo ground truth data. We averaged 30 frames to produce pseudo ground truth.



(a) without BSW loss (b) with BSW loss

Fig. 10. **Visualization of query features and memory items trained with/without BSW loss with t-SNE [76].** For this visualization, we randomly sampled query features from real rain image. The features and memory items are shown in points and stars, respectively. The points with the same color are mapped to the same item. The BSW loss enabled the separation of the items, recording the diverse prototypes of rain streaks. Owing to the BSW loss, the features were highly discriminative and similar rain streaks were clustered well (best viewed in color).



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

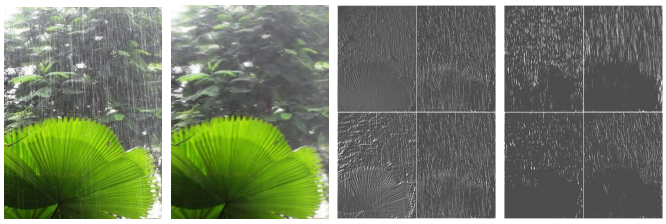
Fig. 11. **Visualization of de-raining result and memory features trained with/without background selective loss.** (a) Input rain image, (b) de-raining result, visualized feature maps from the decoder (c) trained without BSW loss and (d) trained with BSW loss.

TABLE VI
QUANTITATIVE EVALUATION OF THE PROPOSED METHOD TRAINED ON REALDATASET AND TIMELAP, RESPECTIVELY. NOTE THAT (·) DENOTES THE NUMBER OF THE SCENE.

RealDataset	TimeLap	PSNR	SSIM
✓(170)		41.43	0.987
	✓(170)	41.49	0.987
	✓(186)	41.56	0.989
✓(170)	✓(186)	42.12	0.990

TABLE VII
COMPARISON OF RUN TIME (S) AND THE NUMBER OF PARAMETERS. NOTE THAT SPANET [12] USES THE CUPY LIBRARY AND THUS CAN ONLY BE RUN ON A GPU.

	DSC [20]	GMM [20]	JCAS [24]	DDN [8]	DID [9]	JORDER-E [10]	PReNet [16]	SIRR [25]	SPANet [12]	RCDNet [18]	MOSS [27]	Ours
CPU	198.32	681.81	587.46	3.21	77.24	207.03	95.66	3.51	–	34.55	10.31	2.86
GPU	–	–	–	0.34	0.77	1.74	0.69	0.47	0.43	0.57	0.96	0.82
Params.	–	–	–	58.2K	0.37M	4.17M	0.17M	0.18M	0.28M	3.17M	2.86M	0.81M



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

Fig. 12. **Visualization of de-raining and decoder features.** (a) Input rain image, (b) de-raining result, four intermediate feature maps (c) in the encoder of the first convolution layers , and (d) from the decoder of the last convolution layers.

with and without the background selective whitening loss, as shown in Fig. 10. Specifically, we project the embedded content features from the test images into 2D space using t-SNE [76]. The color indicates the memory items, which means that points with the same color are mapped to the same item. The BSW loss was effective in separating and clustering the feature semantically. Therefore, it enhances the diversity and discriminative power of our memory items. With the BSW loss, our method can represent various types of rain streaks.

We further observed the effect of the background selective whitening loss shown in Fig. 11. When our model was trained without $\mathcal{L}_w$, it was difficult to discriminate rain and background, as shown in Fig. 11 (c), whereas our model trained with $\mathcal{L}_w$ shows that the background information was removed, thus the rain streaks are captured in the memory only in Fig. 11 (d). This demonstrates that $\mathcal{L}_w$ helps to estimate rain effectively, and yields improved de-raining performance.

Table V shows the evaluation of our model trained with memory networks in terms of various loss functions. The model trained with a BSW loss achieves better results than the model trained without the BSW loss, demonstrating the BSW loss helps de-raining.

3) Analysis of Using Time-lapse Data: We conducted an experiment to compare the performance of the datasets according to the provided time-lapse benchmark [6], [12]. Because **RealDataset** provides up to 170 scenes, for a fair experimental setup, we also selected 170 scenes of **TimeLab** randomly. Both data utilized two input pairs sampled from 30 images as training data in 170 scenes (*i.e.,* $170 \times {}_{30}C_2$) for training. Table VI shows that the results trained with each dataset showed similar performance improvements because both datasets were constructed in real-world environments. Furthermore, we used the **TimeLab** to compare an experiment by varying the variety of scenes such that 170 scenes and 186 scenes (the total amount of data provided by **TimeLab**). Moreover, we used both datasets for training. From the results, we expect that the proposed method can achieve performance improvements when the time-lapse data contain various scenes.

F. Visualization of Learned Features

In this section, we present the results of a visualization of learned features within our proposed networks. Fig. 12 shows a real rain image, a de-rained result, and visualizations of the learned feature maps of the first and last convolution layers. Fig. 12 (c) shows four intermediate feature maps in the encoder of the first convolution layers. It may clearly be observed that the first convolution seems to calculate the image gradients because the texture details of the leaf that were uncorrelated to the rain streaks are preserved. This indicates that the shallow layers mainly capture the image details extracted from the input rain image. In contrast, the feature maps of the last convolution layer presented in Fig. 12 (d) showed a high correlation with rain streaks, and contained various rain streaks. To summarize, this visualization demonstrates that the de-raining networks effectively estimate the rain streaks and whiten the background, generating plausible de-raining results.

G. Analysis of Run Time and Parameters

We provide data from the running time comparisons of our method with different existing methods in Table VII. The running times were averaged over 100 images with a size of 1000×1000 for evaluation. The hand–crafted methods [20], [20], [24] were run on the CPU according to the provided code, whereas other CNN–based methods were tested on both CPU and GPU. Our method shows GPU runtime similar to that of other other CNNs-based methods, and is much faster than most deep models on the CPU. Although the proposed method leverages the memory network, our model achieved improved de-raining results with comparable computational time.

V. CONCLUSION

We have presented a novel architecture for single image de-raining combining the time-lapse data and the memory network that can handle the various rain streak feature representations, and fully exploit long-term rain streak information. We have also proposed a new background selective whitening (BSW) loss function designed to train a memory network to capture only rain streaks effectively by removing the consistent background information from the time-lapse data. The ablation studies clearly demonstrate the effectiveness of each component and the loss function in our framework. Extensive experiments also show that the proposed method achieved an improved generalization ability on both real-world and synthetic data. In future work, we will explore video de-raining by leveraging a memory network.

Acknowledgments This research was supported by the Yonsei University Research Fund of 2021 (2021-22-0001).

REFERENCES

[1] L. W. Kang, C. W. Lin, and Y. H. Fu, "Automatic single-image-based rain streaks removal via image decomposition," in *IEEE Trans. Image Process.,* vol. 21, pp. 1742–1755, 2011.

[2] X. Fu, B. Liang, Y. Huang, X. Ding, and J. Paisley, "Lightweight pyramid networks for image deraining," in *IEEE Trans. on Neur. Net. Lear.*, vol. 31, pp. 1794–1807, 2019.

[3] S. Li, I. B. Araujo, W. Ren, Z. Wang, E. K. Tokuda, R. H. Junior, R. CesarJunior, J. Zhang, X. Guo, and X. Cao, "Single image deraining: A comprehensive benchmark analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.

[4] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. WH. Lau, and MH. Yang "Vital: Visual tracking via adversarial learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.

[5] K. Jiang, Z. Wang, P. Yi, C. Chen, H. Chen, B. Huang, Y. Luo, J. Ma, and J. Jiang, "Multi-scale progressive fusion network for single image deraining," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020.

[6] J. Cho, S. Kim, D. Min, and K. Sohn, "Single Image Deraining Using Time-Lapse Data," in *IEEE Trans. Image Process.*, vol. 29, pp. 7274–7289, 2020.

[7] H. Zhang, V. Sindagi, and V. M. Patel, "Image de-raining using a conditional generative adversarial network," in *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, pp. 3943–3956, 2019.

[8] X. Fu, J. Huang, D. Zeng, Y. Huang, X. Ding, and J. Paisely, "Removing rain from single images via a deep detail network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017.

[9] H. Zhang and V. M. Patel, "Density-aware single image de-raining using a multi-stream dense network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.

[10] W. Yang, R. T. Tan, J. Feng, Z. Guo, S. Yan, and J. Liu, "Joint rain detection and removal from a single image with contextualized deep networks," in *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, pp. 1377–1393, 2019..

[11] W. Yang, R. T. Tan, J. Feng, J. Liu, Z. Guo, and S. Yan, "Deep joint rain detection and removal from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017.

[12] T. Wang, X. Yang, K. Xu, S. Chen, Q. Zhang, and R. W. Lau, "Spatial Attentive Single-Image Deraining with a High Quality Real Rain Dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.

[13] W. Yang, S. Wang, and J. Liu, "Removing Arbitrary-Scale Rain Streaks via Fractal Band Learning With Self-Supervision," in *IEEE Trans. Image Process.*, vol. 29, pp. 6759–6772, 2020.

[14] 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 Int. Conf. Multimedia*, 2018.

[15] X. Fu, J. Huang, X. Ding, Y. Liao and J. Paisley, "Clearing the skies: A deep network architecture for single-image rain removal," in *IEEE Trans. Image Process.*, vol. 26, pp. 2944–2956, 2017.

[16] D. Ren, W. Zuo, Q. Hu, P. Zhu, and D. Meng, "Progressive Image Deraining Networks: A Better and Simpler Baseline," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.

[17] W. Yang, J Liu, S Yang, and Z Guo, "Scale-free single image deraining via visibility-enhanced recurrent wavelet learning," in *IEEE Trans. Image Process.*, vol. 28, pp. 2948–2961, 2019.

[18] H. Wang, Q. Xie, Q. Zhao, and D. Meng, "A model-driven deep neural network for single image rain removal," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020.

[19] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, M. H. Yang, and L. Shao, "Multi-Stage Progressive Image Restoration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021.

[20] Y. Luo, Y. Xu, and H. Ji, "Removing rain from a single image via discriminative sparse coding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015.

[21] Y. Li, R. T. Robby, X. Guo, J. Lu, and M. S. Brown, "Rain streak removal using layer priors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016.

[22] Y. Chang, L. Yan, and S. Zhong, "Transformed low-rank model for line pattern noise removal," in *Proc. Int. Conf. Comput. Vis.*, 2017.

[23] L. Zhu, C. W. Fu, D. Lischinski, and P. A. Heng, "Joint bi-layer optimization for single-image rain streak removal," in *Proc. Int. Conf. Comput. Vis.*, 2017.

[24] S. Gu, D. Meng, W. Zuo, and L. Zhang, "Joint convolutional analysis and synthesis sparse representation for single image layer separation," in *Proc. Int. Conf. Comput. Vis.*, 2017.

[25] W. Wei, D. Meng, Q. Zhao, Z. Xu, and Y. Wu, "Semi-supervised Transfer Learning for Image Rain Removal," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.

[26] R. Yasarla, V. A. Sindagi, and V. M. Patel, "Syn2Real transfer learning for image deraining using Gaussian processes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020.

[27] H. Huang, A. Yu and R. He, "Memory Oriented Transfer Learning for Semi-Supervised Image Deraining," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021.

[28] https://www.photoshopessentials.com/photo-effects/rain/.

[29] K.Garg and S. K. Nayar, "Photorealistic rendering of rain streaks," *ACM Trans. on Grap.*, vol. 25, pp. 996–1002, 2006.

[30] W. Yang, R. T. Tan, S. Wang, Y. Fang, and J. Liu, "Single image deraining: From model-based to data-driven and beyond," in *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, pp. 4059–4077, 2020.

[31] K. Garg and S. K. Nayar, "Detection and removal of rain from videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2004.

[32] K. Garg and S. K. Nayar, "When does a camera see rain?," in *Proc. Int. Conf. Comput. Vis.*, 2005.

[33] K. Garg and S. K. Nayar, "Vision and rain," in *IEEE Int. J. Comput. Vis.*, vol. 75, pp. 3–27, 2007.

[34] X. Zhang, H. Li, Y. Qi, W. K. Leow, and T. K. Ng, "Rain removal in video by combining temporal and chromatic properties," in *IEEE Int. Conf. on Multi. and Expo.*, 2006.

[35] P. Barnum, T. Kanade, and S. Narasimhan, "Spatio-temporal frequency analysis for removing rain and snow from videos," in *Proc. of Int. Photometric Anal. Comput. Vis.* 2007.

[36] J. Kim, J. Sim, and C. Kim, "Video deraining and desnowing using temporal correlation and low-rank matrix completion," in *IEEE Trans. Image Process.*, vol. 24, pp. 2658–2670, 2015.

[37] T. X. Jiang, T. Z. Huang, X. L. Zhao, L. J. Deng, and Y. Wang, "A novel tensor-based video rain streaks removal approach via utilizing discriminatively intrinsic priors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017.

[38] W. Ren, J. Tian, Z. Han, A. Chan, and Y. Tang, "Video desnowing and deraining based on matrix decomposition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017.

[39] W. Wei, L. Yi, Q. Xie, Q. Zhao, D. Meng, and Z. Xu, "Should we encode rain streaks in video as deterministic or stochastic?," in *Proc. Int. Conf. Comput. Vis.*, 2017.

[40] M. Li, Q. Xie, Q. Zhao, W. Wei, S. Gu, J. Tao, and D. Meng, "Video rain streak removal by multiscale convolutional sparse coding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.

[41] J. Chen, C. H. Tan, J. Hou, L. P. Chau, and H. Li, "Robust video content alignment and compensation for rain removal in a cnn framework," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.

[42] J. Liu, W. Yang, S. Yang, and Z. Guo, "Erase or fill? deep joint recurrent rain removal and reconstruction in videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018.

[43] J. Liu, W. Yang, S. Yang, and Z. Guo, "D3r-net: Dynamic routing residue recurrent network for video rain removal," in *IEEE Trans. Image Process.*, 2018.

[44] W. Yang, J. Liu, and J. Feng, "Frame-consistent recurrent video deraining with dual-level flow," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.

[45] X. Li, J. Wu, Z. Lin, H. Liu, and H. Zha, "Recurrent squeeze-and-excitation context aggregation net for single image deraining," in *Eur. Conf. on Comput. Vis.*, 2018.

[46] X. Hu, C. W. Fu, L. Zhu, and P. A. Heng, "Depth-attentional features for single-image rain removal," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.

[47] X. Hu, L. Zhu, T. Wang, C. W. Fu, and P. A. Heng, "Single-Image Real-Time Rain Removal Based on Depth-Guided Non-Local Features," in *IEEE Trans. Image Process.*, vol. 30, pp. 1759–1770, 2020.

[48] C. Y. Lin, Z. Tao, A. S. Xu, L. W. Kang, and F. Akhyar, "Sequential dual attention network for rain streak removal in a single image," in *Trans. Image Process.*, vol. 29, pp. 9250–9265, 2020.

[49] G. Wang, C. Sun, and A. Sowmya, "Context-enhanced representation learning for single image deraining," in *Int. J. Comput. Vis.*, vol. 129, no. 5, pp. 1650–1674, 2021.

[50] S. Li, W. Ren, F. Wang, I. B. Araujo, E. K. Tokuda, R. H. Junior, Z. Wang, and X. Cao, "A Comprehensive Benchmark Analysis of Single Image Deraining: Current Challenges and Future Perspective," in *Int. J. Comput. Vis.*, vol. 129, pp. 1301–1322, 2021.

[51] J. Weston, S. Chopra, and A. Bordes, "Memory networks," in *Adv. Neural Inform. Process. Syst.*, 2015.

[52] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "End-to-end memory networks," in *Int. Conf. Learn. Represent.*, 2015.
[53] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," in *arXiv preprint arXiv:1410.5401*, 2015.
[54] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra and T. Lillicrap, "One-shot learning with memory-augmented neural networks," in *arXiv preprint arXiv:1605.06065*, 2016.
[55] S. Na, S. Lee, J. Kim and G. Kim, "A read-write memory network for movie story understanding," in *Proc. Int. Conf. Comput. Vis.*, 2017.
[56] H. Seong, J. Hyun and E. Kim, "Kernelized Memory Network for Video Object Segmentation," in *Eur. Conf. on Comput. Vis.*, 2020.
[57] S. Oh, J. Lee, N. Xu, and S. Kim, "Video object segmentation using space-time memory networks," in *Proc. Int. Conf. Comput. Vis.*, 2019.
[58] X. Lu, W. Wang, M. Danelljan, T. Zhou, J. Shen, and L. V. Gool, "Video object segmentation with episodic graph memory networks," in *Eur. Conf. on Comput. Vis.*, 2020.
[59] M. Zhu, P. Pan, W. Chen, and Y. Yang, "Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.
[60] C. Fan, X. Zhang, S. Zhang, W. Wang, C. Zhang, and H. Huang, "Heterogeneous memory enhanced multimodal attention model for video question answering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.
[61] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. V. D. Hengel, "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," in *Proc. Int. Conf. Comput. Vis.*, 2019.
[62] H. Park, J. Noh, and B. Ham, "Learning memory-guided normality for anomaly detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020.
[63] R. Yasarla and V. M. Patel, "Confidence measure guided single image de-raining," in *IEEE Trans. Image Process.*, vol. 29, pp. 4544–4555, 2020.
[64] S. Nam, C. Ma, M. Chai, W. Brendel, N. Xu, and S. Kim, "End-to-end time-lapse video synthesis from a single outdoor image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.
[65] I. Anokhin, P. Solovev, D. Korzhenkov, A. Kharlamov, T. Khakhulin, A. Silvestrov, S. Nikolenko, V. Lempitsky, and G. Sterkin, "High-resolution daytime translation without domain labels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020.
[66] C. C. Cheng, H. Y. Chen, and W. C. Chiu, "Time flies: Animating a still image with time-lapse video as reference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020.
[67] W. C. Ma, H. Chu, B. Zhou, R. Urtasun, and A. Torralba, "Single image intrinsic decomposition without a single intrinsic image," in *Eur. Conf. on Comput. Vis.*, 2018.
[68] L. Lettry, K. Vanhoey, L. Van Gool, "Unsupervised Deep Single-Image Intrinsic Decomposition using Illumination-Varying Image Sequences," in *Comput. Grap. Forum*, 2018.
[69] L. Lettry, K. Vanhoey, L. Van Gool, "Deep unsupervised intrinsic image decomposition by Siamese training," in *arXiv preprint arXiv:1803.00805*, 2018.
[70] S. Choi, S. Jung, H. Yun, J. Kim, S. Kim and J. Choo, "RobustNet: Improving Domain Generalization in Urban-Scene Segmentation via Instance Selective Whitening," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020.
[71] A. Paszke, S. Gross, S. Chintala, G.Chanan, E. Yang, Z. DeVito, ZLA. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.
[72] D. P. Kingma, and J. Ba, "A method for stochastic optimization," in *arXiv preprint arXiv:1412.6980*, 2014.
[73] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," in *IEEE Trans. Image Process.*, vol. 15, pp. 430–444, 2006.
[74] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "FSIM: A feature similarity index for image quality assessment," in *IEEE Trans. Image Process.*, vol. 20, pp. 2378–2386, 2011.
[75] I. Loshchilov and F. Hutter , "Sgdr: Stochastic gradient descent with warm restarts," in *arXiv preprint arXiv:1608.03983*, 2016.
[76] L. Van Der Maaten, "Accelerating t-SNE using tree-based algorithms," in *The Journal of Machine Learning Research*, 2014.

Jaehoon Cho received the B.S. degree in electronic engineering and avionics from Korea Aerospace University, Gyeonggi, Korea, in 2016. He received the Ph.D. degrees from the School of Electrical and Electronic Engineering from Yonsei University, Seoul, Korea, in 2022. Since 2022, he has been a senior research engineer with the with the Autonomous Driving SW Development Team 1, Hyundai Motor Group. His current research interests include deep-learning based image processing, particularly in the area of bad weather restoration, related applications.

Seungryong Kim received the B.S. and Ph.D. degrees from the School of Electrical and Electronic Engineering from Yonsei University, Seoul, Korea, in 2012 and 2018, respectively. From 2018 to 2019, he was Post-Doctoral Researcher in Yonsei University, Seoul, Korea. From 2019 to 2020, he has been Post-Doctoral Researcher in School of Computer and Communication Sciences at École Polytechnique Féd érale de Lausanne (EPFL), Lausanne, Switzerland. Since 2020, he has been an assistant professor with the Department of Computer Science and Engineering, Korea University, Seoul. His current research interests include 2D/3D computer vision, computational photography, and machine learning.

Kwanghoon Sohn (M'92-SM'12) received the B.E. degree in electronic engineering from Yonsei University, Seoul, Korea, in 1983, the M.S.E.E. degree in electrical engineering from the University of Minnesota, Minneapolis, MN, USA, in 1985, and the Ph.D. degree in electrical and computer engineering from North Carolina State University, Raleigh, NC, USA, in 1992. He was a Senior Member of the Research engineer with the Satellite Communication Division, Electronics and Telecommunications Research Institute, Daejeon, Korea, from 1992 to 1993, and a Post-Doctoral Fellow with the MRI Center, Medical School of Georgetown University, Washington, DC, USA, in 1994. He was a Visiting Professor with Nanyang Technological University, Singapore, from 2002 to 2003. He is currently an Underwood Distinguished Professor with the School of Electrical and Electronic Engineering, Yonsei University. His research interests include 3D image processing and computer vision.