Generative Status Estimation and Information Decoupling for Image Rain Removal

Di Lin[1,†], **Xin Wang**[2,†], **Jia Shen**[1], **Renjie Zhang**[2], **Ruonan Liu**[1], **Miaohui Wang**[3], **Wuyuan Xie**[3], **Qing Guo**[4], and **Ping Li**[2,*]

[1]Tianjin University, China
[2]The Hong Kong Polytechnic University, Hong Kong
[3]Shenzhen University, China
[4]Center for Frontier AI Research, A*STAR, Singapore
p.li@polyu.edu.hk

Abstract

Image rain removal requires the accurate separation between the pixels of the rain streaks and object textures. But the confusing appearances of rains and objects lead to the misunderstanding of pixels, thus remaining the rain streaks or missing the object details in the result. In this paper, we propose **SEIDNet** equipped with the generative **Status Estimation** and **Information Decoupling** for rain removal. In the status estimation, we embed the pixel-wise statuses into the status space, where each status indicates a pixel of the rain or object. The status space allows sampling multiple statuses for a pixel, thus capturing the confusing rain or object. In the information decoupling, we respect the pixel-wise statuses, decoupling the appearance information of rain and object from the pixel. Based on the decoupled information, we construct the kernel space, where multiple kernels are sampled for the pixel to remove the rain and recover the object appearance. We evaluate SEIDNet on the public datasets, achieving state-of-the-art performances of image rain removal. The experimental results also demonstrate the generalization of SEIDNet, which can be easily extended to achieve state-of-the-art performances on other image restoration tasks (e.g., snow, haze, and shadow removal). We release the implementation of SEIDNet via https://github.com/wxxx1025/SEIDNet.

1 Introduction

Image rain removal relies on the understanding of the appearances of the rains and the objects. Most of the current methods employ the discriminative network to learn the visual features of the pixels, for representing the visual appearances of the rain streaks and object textures. Based on the visual features, the network learns the shared [1, 2, 3, 4, 5, 6, 7] or dynamic convolutional kernels [8, 9] for rain removal on the pixels. Intuitively, the kernels for rain removal respect the pixel-wise statuses (i.e., rain or object), reducing the rainy intensities and recovering the object details of the pixels.

The challenge of rain removal mainly stems from the fact that some of the rains are similar to the objects (see the input regions in Figure 1(a)). The confusion between the rains and the objects is inevitably encoded into the visual features. It yields the inappropriate kernels for the pixels, contributing to the erroneous results like leaving the rain streaks and removing the object textures in the image (see the predictions in the pink rectangles of Figure 1(b)). The discriminative network provides a deterministic kernel for a pixel. But it loses the chance of computing more appropriate

† Di Lin and Xin Wang contributed equally to this work.
* Ping Li is the corresponding author.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

https://doi.org/10.52202/068431-0333

(a) Rainy Image (b) Result of Rain Removal

Figure 1: In the left-most column, the input regions of the rainy image (a) have similar rain streaks and object textures. The top/bottom pair of input regions are processed by the similar kernels. In the result (b), these kernels yield good predictions in the blue rectangles, but remaining rains and missing textures in the pink rectangles. Here, we employ EfDeRain [8] to estimate the kernels.

kernels for a misunderstood pixel. The popular methods [2, 3, 4, 10, 11, 12, 13, 14, 15, 16, 17, 18] use different kernels to process the rains at several stages. Yet, the confusing information at the early stages still misleads the kernel computation at the later stages.

In this paper, we present a novel approach for capturing the confusing information of the pixels. The key idea is to use the generative network to learn the probability distribution of the pixel-wise status and the dynamic convolutional kernel. The distribution captures the correlation between status and kernel. Given the distribution, we regard the feature of the pixel as the condition, generating multiple statuses and kernels. The statuses facilitate more focused learning of the dynamic kernels, which capture the rain and object characteristics. Compared to single status, multiple statuses can be used for generating more kernels, which provide more opportunities for refining the result of rain removal.

We construct **SEIDNet**, a generative network equipped with the pixel-wise **Status Estimation** and **Information Decoupling** for rain removal. We illustrate the construction of SEIDNet in Figure 2. We use the rainy image and the object layer[1] as the training sample. In the status estimation (see Figure 2(a)), we subtract the object layer from the rainy image, achieving the statuses of all pixels in the rainy image. We construct a conditional variational auto-encoder (CVAE) to embed the pixel-wise statuses of different pixels into the status space. With the status space, we use the feature of the pixel as the condition to generate multiple statuses that capture the confusing appearance of the pixel.

In the information decoupling (see Figure 2(b)), we use the statuses to yield the feature maps that represent the decoupled information of rain and object. We exploit these feature maps to learn the kernels. We employ another CVAE to embed the learned kernels into the kernel space, where we use the feature and the status of the pixel as the condition to generate the kernel. The status estimation can generate multiple statuses, helping to generate multiple kernels. These kernels effectively reduce the confusing information of the pixel, finally producing a better result of rain removal.

We evaluate SEIDNet on the public datasets for rain removal (i.e., Rain100H [16], Rain100L [16], Rain1400 [19], Rain13K [10] and SPA [3]), achieving state-of-the-art performances. Furthermore, we extend SEIDNet to various image restoration tasks (e.g., snow, haze, and shadow removal), where SEIDNet also surpasses the recent methods on the public datasets [20, 21, 22].

2 Related work

2.1 Discriminative Networks

The recent methods use the deep discriminative network to learn the visual features of rain and object. Zhang et al. [4] propose a residual-aware classifier to recognize the rain density. Yang et al. [16] use large convolutional kernels to capture rich visual information for recovering the details of the dense rain streaks and object textures. Zhang et al. [23], Wang et al. [3], and Li et al. [13] use the

[1] We refer to the object layer as the image without rain.

https://doi.org/10.52202/068431-0333

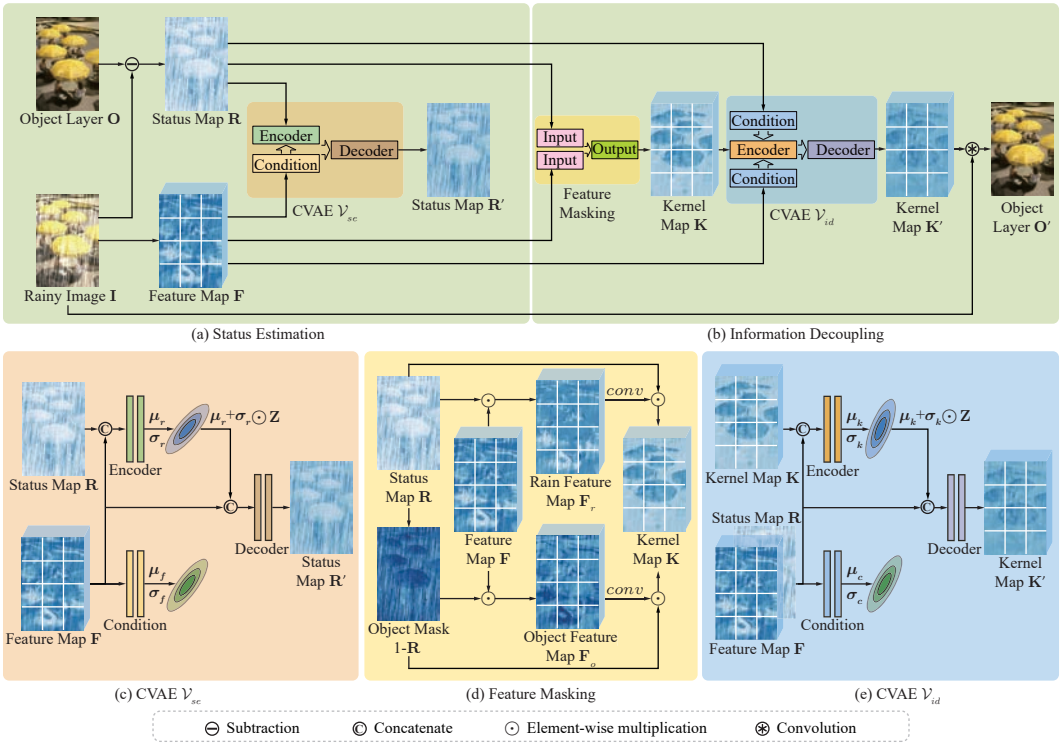


Figure 2: The training architecture of SEIDNet. The architecture has the (a) status estimation and (b) information decoupling. We use (c) the CVAE for learning the status space, (d) the feature masking for yielding the rain and object feature maps, and (e) the CVAE for learning the kernel space.

multi-scale information to capture the appearances of rains and objects in local and global ranges. Li et al. [12] use the dilated convolution to capture the multi-scale image information for predicting the diverse shapes of the rain streaks. Wang et al. [2] propose the convolutional dictionary model, where a set of rain kernels are used for capturing the appearances of the rain streaks and object textures.

The discriminative networks are also used for learning the correlation between rain and object in terms of their appearances. Luo et al. [24] combine the sparse coding and the greedy pursuit algorithms to separate the rain and object layers. Li et al. [25] utilize Gaussian Mixture Models (GMMs) to involve the patch-based priors of rain and object. Deng et al. [26] use a unified network for rain removal and repair of object textures. Fu et al. [19] propose the detail layer of the rain streaks and object contours, for reducing the impact of low-frequency object textures on the rain removal.

Based on the pixel-wise features, the discriminative networks compute the shared [1, 2, 3, 4, 5, 6, 7] or the dynamic convolutional kernels [8, 9] for rain removal on the pixels. Yet, the similar appearances of the rains and the objects let the discriminative network predict the problematic kernels. In contrast, we use the generative network that learns the probability distribution of the pixel-wise status. With the distribution, we sample multiple statuses for capturing the confusing information of the pixel. It assists the computation of kernels for processing the pixels of the rain streaks and object textures.

2.2 Generative Networks

The generative networks, such as the variational auto-encoders (VAEs) [27, 28, 29, 30] and the generative adversarial networks (GANs) [31, 32, 33, 34, 35], have been widely used for image generation. Some of the methods employ the generative networks to recover the object layer of the rainy image. Li et al. [14] input the rainy image to the conditional GAN that preserves the object textures. Rui et al. [17] resort to the visual attention mechanism, attending to the important image regions with rain streaks. Zhang et al. [23] propose the delicate loss function to alleviate the artifacts generated by GAN in the object layer. Du et al. [36] equip the conditional VAE [37] to the spatial density estimation of the rain streaks for more accurate rain removal.

The existing methods use a stand-alone generative network for embedding the features of the rains and objects into the latent space. But the confusing rains and objects may mislead the construction of

https://doi.org/10.52202/068431-0333

the latent space. These methods may achieve the latent vector of the rain (or object) from the latent space, where the vector mistakenly represents the object (or rain). Instead, we utilize a couple of CVAEs, for learning the probability distribution of the pixel-wise status and dynamic kernel. We use the first CVAE to sample multiple statuses from the latent space, capturing the confusing information of the pixel. With the second CVAE, we use multiple statuses to sample the kernels. These kernels are learned from the decoupled information of the rains and objects. They better reduce the confusing information and refine the pixel intensities in the result.

3 Method Overview

We introduce the probability distribution of the pixel-wise status and kernel. The distribution is learned by CVAEs in the status estimation and information decoupling of SEIDNet.

Probability Distribution For the rainy image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$, we compute the visual feature map $\mathbf{F} \in \mathbb{R}^{H\times W\times C}$ and the status map $\mathbf{R} \in \mathbb{R}^{H\times W}$. $\mathbf{F}(x, y) \in \mathbb{R}^C$ represents the feature vector of the pixel located at (x, y) in the rainy image. $\mathbf{R}(x, y) \in \mathbb{R}$ is the score for the pixel. A higher (or lower) score indicates the status of the rain (or object). We compute the kernel map $\mathbf{K} \in \mathbb{R}^{H\times W\times (S\times S\times C)}$, where $\mathbf{K}(x, y) \in \mathbb{R}^{S\times S\times C}$ is the $S \times S$ kernel with C channels for the pixel at (x, y). We construct the generative network to learn the probability distribution $P(\mathbf{R}, \mathbf{K} \mid \mathbf{F})$, which takes condition as the feature map $\mathbf{F}$ for computing the status map $\mathbf{R}$ and the kernel map $\mathbf{K}$. We formulate $P(\mathbf{R}, \mathbf{K} \mid \mathbf{F})$ as:

$$P(\mathbf{R}, \mathbf{K} \mid \mathbf{F}) = \int P(\mathbf{R}, \mathbf{K} \mid \mathbf{F}, \mathbf{Z})\ P(\mathbf{Z})\ d\mathbf{Z}. \tag{1}$$

$\mathbf{Z} \in \mathbb{R}^{H\times W\times C}$ is a set of latent variables generated by the normal distribution $P(\mathbf{Z})$. We factorize the distribution $P(\mathbf{R}, \mathbf{K} \mid \mathbf{F}, \mathbf{Z})$ into two distributions, $P(\mathbf{K} \mid \mathbf{R}, \mathbf{F}, \mathbf{Z})$ and $P(\mathbf{R} \mid \mathbf{F}, \mathbf{Z})$, as:

$$P(\mathbf{R}, \mathbf{K} \mid \mathbf{F}, \mathbf{Z}) = P(\mathbf{K} \mid \mathbf{R}, \mathbf{F}, \mathbf{Z})\ P(\mathbf{R} \mid \mathbf{F}, \mathbf{Z}). \tag{2}$$

The factorization of probability distribution enables a more focused learning of the dynamic kernel based on the pixel-wise status. We use two CVAEs in the status estimation and information decoupling to learn the conditional distributions $P(\mathbf{R} \mid \mathbf{F}, \mathbf{Z})$ and $P(\mathbf{K} \mid \mathbf{R}, \mathbf{F}, \mathbf{Z})$, respectively.

Status Estimation As illustrated in Figure 2(a), we use the rainy image $\mathbf{I}$ and the object layer $\mathbf{O} \in \mathbb{R}^{H\times W\times 3}$ to train the CVAE $\mathcal{V}_{se}$. $\mathcal{V}_{se}$ learns the distribution $P(\mathbf{R} \mid \mathbf{F}, \mathbf{Z})$, for modeling the status space. We subtract $\mathbf{O}$ from $\mathbf{I}$, to achieve the status map $\mathbf{R}$. $\mathcal{V}_{se}$ embeds the statuses in $\mathbf{R}$ into the status space. As illustrate in Figure 2(c), $\mathcal{V}_{se}$ regards the feature map $\mathbf{F}$ as the condition, for sampling the status map $\mathbf{R}'$ from the status space. We regard $\mathbf{R}'$ as the estimation of $\mathbf{R}$.

Information Decoupling We use the status map $\mathbf{R}$ and the feature map $\mathbf{F}$ to train the CVAE $\mathcal{V}_{id}$ in the information decoupling (see Figure 2(b)). $\mathcal{V}_{id}$ learns the distribution $P(\mathbf{K} \mid \mathbf{R}, \mathbf{F}, \mathbf{Z})$ that models the kernel space. First, we feed $\mathbf{R}$ and $\mathbf{F}$ into the feature masking (see Figure 2(d)). We decouple $\mathbf{F}$ into the rain feature map $\mathbf{F}_r \in \mathbb{R}^{H\times W\times C}$ and the object feature map $\mathbf{F}_o \in \mathbb{R}^{H\times W\times C}$. We use $\mathbf{F}_r$ and $\mathbf{F}_o$ to compute the kernel map $\mathbf{K}$ that removes the rains and recovers the objects. Next, $\mathcal{V}_{id}$ embeds $\mathbf{K}$ into the kernel space. As illustrated in Figure 2(e), $\mathcal{V}_{id}$ regards $\mathbf{F}$ and $\mathbf{R}$ as the condition, for sampling the kernel map $\mathbf{K}'$ from the kernel space. $\mathbf{K}'$ estimates $\mathbf{K}$ for rain removal.

4 Architecture of SEIDNet

Below, we introduce the training and testing architectures of SEIDNet for rain removal.

4.1 Training Architecture

Status Estimation We illustrate the training architecture of SEIDNet in Figure 2. In the status estimation (see Figure 2(a)), we achieve the status map $\mathbf{R} \in \mathbb{R}^{H\times W}$ as:

$$\mathbf{R} = \sigma(conv(\mathbf{I} - \mathbf{O})). \tag{3}$$

where $\mathbf{R}(x, y) \in [0, 1]$. σ and $conv$ are the sigmoid function and a set of convolutional layers. For the rainy image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$, we use the convolution to learn the feature map $\mathbf{F} \in \mathbb{R}^{H\times W\times C}$. We

https://doi.org/10.52202/068431-0333

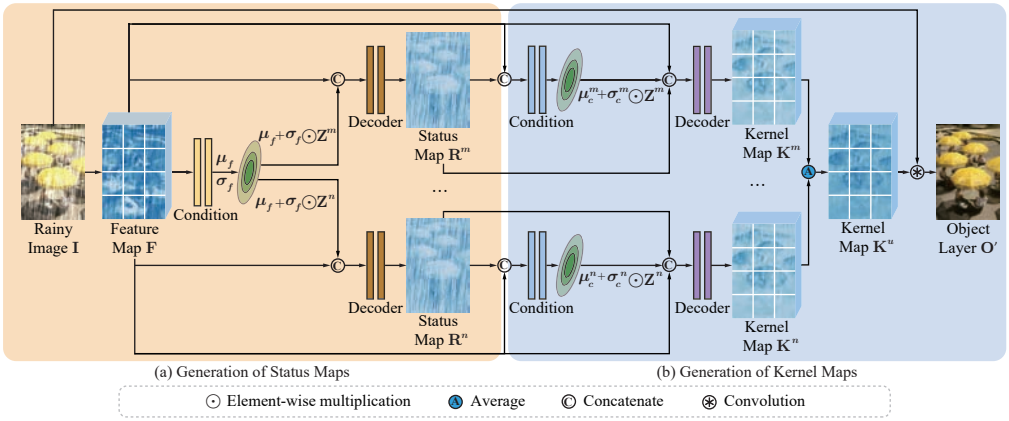


Figure 3: The testing architecture of SEIDNet. It contains the condition and decoder branches of CVAEs, for generating (a) status maps and (b) kernel maps. For brevity, we only illustrate two pairs of status and kernel maps, i.e., $(\mathbf{R}^m, \mathbf{K}^m)$ and $(\mathbf{R}^n, \mathbf{K}^n)$. More maps can be generated for testing.

input $\mathbf{R}$ and $\mathbf{F}$ to the CVAE $\mathcal{V}_{se}$ (see Figure 2(c)), whose encoder, condition, and decoder branches are formulated as:

$$[\boldsymbol{\mu}_r,\ \boldsymbol{\sigma}_r] = encoder([\mathbf{R},\ \mathbf{F}]),\ \ [\boldsymbol{\mu}_f,\ \boldsymbol{\sigma}_f] = condition(\mathbf{F}),\ \ \mathbf{R}' = decoder([\mathbf{F},\ \boldsymbol{\mu}_r + \boldsymbol{\sigma}_r \odot \mathbf{Z}]). \quad (4)$$

$[\cdot]$ is the feature concatenation along the channel dimension. $\odot$ is the element-wise multiplication.

The encoder and condition branches take input as $[\mathbf{R},\ \mathbf{F}]$ and $\mathbf{F}$, yielding the mean value maps $\boldsymbol{\mu}_r, \boldsymbol{\mu}_f \in \mathbb{R}^{H\times W\times C}$ and the standard deviation maps $\boldsymbol{\sigma}_r, \boldsymbol{\sigma}_f \in \mathbb{R}^{H\times W\times C}$. We use the normal distribution $\mathcal{N}(0,1)$ to generate the latent variable map $\mathbf{Z} \in \mathbb{R}^{H\times W\times C}$. We input $\mathbf{F}$, $\mathbf{Z}$ and $(\boldsymbol{\mu}_r, \boldsymbol{\sigma}_r)$ to the decoder, yielding the status map $\mathbf{R}' \in \mathbb{R}^{H\times W}$. We use the sigmoid function to normalize $\mathbf{R}'$.

During the network training, we use L_2-norm (denoted as L_2) to penalize the difference between the status map $\mathbf{R}$ and the estimated counterpart $\mathbf{R}'$. Besides, we compute Kullback-Leibler Divergence (denoted as KL) between the Gaussian distributions $\mathcal{G}(\boldsymbol{\mu}_r, \boldsymbol{\sigma}_r)$ and $\mathcal{G}(\boldsymbol{\mu}_f, \boldsymbol{\sigma}_f)$. It allows to use $(\boldsymbol{\mu}_f, \boldsymbol{\sigma}_f)$ in place of $(\boldsymbol{\mu}_r, \boldsymbol{\sigma}_r)$ that are unavailable during testing. We define $\alpha = 4$ as the weight of KL divergence. L_2-norm and KL divergence compose the status estimation loss L_{se} as:

$$L_{se} = L_2(\mathbf{R},\ \mathbf{R}') + \alpha KL(\mathcal{G}(\boldsymbol{\mu}_r, \boldsymbol{\sigma}_r)\ ||\ \mathcal{G}(\boldsymbol{\mu}_f, \boldsymbol{\sigma}_f)). \quad (5)$$

This loss guides $\mathcal{V}_{se}$ to learn the distribution $P(\mathbf{R} \mid \mathbf{F}, \mathbf{Z})$ in Eq. (2), which models the status space.

Information Decoupling In the information decoupling (see Figure 2(b)), we pass the status map $\mathbf{R}$ and the feature map $\mathbf{F}$ into the feature masking (see Figure 2(d)), achieving the feature maps $\mathbf{F}_r, \mathbf{F}_o \in \mathbb{R}^{H\times W\times C}$ in Eq. (6). Based on $\mathbf{F}_r, \mathbf{F}_o$ that respectively represent the appearances of rains and objects in the image $\mathbf{I}$, we compute the kernel maps $\mathbf{K}_r, \mathbf{K}_o \in \mathbb{R}^{H\times W\times (S\times S\times C)}$.

$$\mathbf{F}_r = \mathbf{R} \odot \mathbf{F},\ \ \mathbf{K}_r = conv(\mathbf{F}_r), \quad \mathbf{F}_o = (1-\mathbf{R}) \odot \mathbf{F},\ \ \mathbf{K}_o = conv(\mathbf{F}_o). \quad (6)$$

We use the status map $\mathbf{R}$ to weight $\mathbf{K}_r$ and $\mathbf{K}_o$, yielding the kernel map $\mathbf{K} \in \mathbb{R}^{H\times W\times (S\times S\times C)}$ as:

$$\mathbf{K} = \mathbf{R} \odot \mathbf{K}_r + (1-\mathbf{R}) \odot \mathbf{K}_o. \quad (7)$$

As illustrated in Figure 2(e), the CVAE $\mathcal{V}_{id}$ in the information decoupling also has the encoder, condition and decoder branches, which are formulated as:

$$[\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k] = encoder([\mathbf{K}, \mathbf{F}, \mathbf{R}]),\ \ [\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c] = condition([\mathbf{F}, \mathbf{R}]),\ \ \mathbf{K}' = decoder([\mathbf{F}, \mathbf{R}, \boldsymbol{\mu}_k + \boldsymbol{\sigma}_k \odot \mathbf{Z}]), \quad (8)$$

We input $[\mathbf{K}, \mathbf{F}, \mathbf{R}]$ and $[\mathbf{F}, \mathbf{R}]$ into the encoder and condition branches, respectively, producing the mean value maps $\boldsymbol{\mu}_k, \boldsymbol{\mu}_c \in \mathbb{R}^{H\times W\times C}$ and the standard deviation maps $\boldsymbol{\sigma}_k, \boldsymbol{\sigma}_c \in \mathbb{R}^{H\times W\times C}$. We feed $[\mathbf{F}, \mathbf{R}, \boldsymbol{\mu}_k + \boldsymbol{\sigma}_k \odot \mathbf{Z}]$ into the decoder, yielding the kernel map $\mathbf{K}' \in \mathbb{R}^{H\times W\times (S\times S\times C)}$.

We resort to the L_2-norm and KL divergence to construct the information decoupling loss L_{id} as:

$$L_{id} = L_2(\mathbf{K},\ \mathbf{K}') + \alpha KL(\mathcal{G}(\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)\ ||\ \mathcal{G}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c)). \quad (9)$$

$\mathcal{V}_{id}$ learns the conditional distribution $P(\mathbf{K} \mid \mathbf{R}, \mathbf{F}, \mathbf{Z})$ in Eq. (2), which models the kernel space.

https://doi.org/10.52202/068431-0333

Deraining and Overall Losses Given the kernel map $\mathbf{K}'$, we perform the convolution on the rainy image $\mathbf{I}$ and estimate the object layer $\mathbf{O}'$. As formulated in the left of Eq. (10), the convolution (denoted as $\circledast$) is pixel-wise. where the pixel $\mathbf{I}(x, y)$ is processed by the kernel $\mathbf{K}'(x, y)$. We use the structural similarity (denoted as $SSIM$) loss and L_2-norm to penalize the difference between the object layer $\mathbf{O}$ and the estimated layer $\mathbf{O}'$, yielding the deraining loss L_{de} in the right of Eq. (10).

$$\mathbf{O}'(x, y) = \mathbf{K}'(x, y) \circledast \mathbf{I}(x, y), \;\; L_{de} = L_2(\mathbf{O}, \; \mathbf{O}') + \beta SSIM(\mathbf{O}, \mathbf{O}'), \tag{10}$$

where $\beta = 0.2$. With the losses L_{se}, L_{id} and L_{de}, we form the overall loss L as:

$$L = L_{se} + L_{id} + L_{de}. \tag{11}$$

4.2 Testing Architecture

Kernel Generation During the network testing, we follow the convention to remove all encoder branches of the CVAEs $\mathcal{V}_{se}$ and $\mathcal{V}_{id}$. It results in the testing architecture (see Figure 3). We formulate the process of using $\mathcal{V}_{se}$ and $\mathcal{V}_{id}$ to generate the status map $\mathbf{R}^m$ and the kernel map $\mathbf{K}^m$ as:

$$\begin{aligned} [\boldsymbol{\mu}_f, \; \boldsymbol{\sigma}_f] = condition(\mathbf{F}), \;\; \mathbf{R}^m = decoder([\mathbf{F}, \; \boldsymbol{\mu}_f + \boldsymbol{\sigma}_f \odot \mathbf{Z}^m]), \\ [\boldsymbol{\mu}_c^m, \; \boldsymbol{\sigma}_c^m] = condition([\mathbf{F}, \; \mathbf{R}^m]), \;\; \mathbf{K}^m = decoder([\mathbf{F}, \; \mathbf{R}^m, \; \boldsymbol{\mu}_c^m + \boldsymbol{\sigma}_c^m \odot \mathbf{Z}^m]). \end{aligned} \tag{12}$$

As illustrated in Figure 3(a), we use the rainy image $\mathbf{I}$ to compute the the visual feature map $\mathbf{F}$. We pass $\mathbf{F}$ into the condition branch of $\mathcal{V}_{se}$, achieving the mean value map $\boldsymbol{\mu}_f$ and the standard deviation map $\boldsymbol{\sigma}_f$. We use the normal distribution to generate the latent variable map $\mathbf{Z}^m$. As formulated in Eq. (12), the decoder of $\mathcal{V}_{se}$ uses $\mathbf{F}$, $\mathbf{Z}^m$ and $(\boldsymbol{\mu}_f, \boldsymbol{\sigma}_f)$ to generate the status map $\mathbf{R}^m$.

As illustrated in Figure 3(b), we use the condition branch of $\mathcal{V}_{id}$, which takes input as $\mathbf{F}$ and $\mathbf{R}^m$, to compute the mean value map $\boldsymbol{\mu}_c^m$ and the standard deviation map $\boldsymbol{\sigma}_c^m$. As formulated in Eq. (12), the decoder of $\mathcal{V}_{id}$ uses $\mathbf{F}$, $\mathbf{R}^m$, $\mathbf{Z}$ and $(\boldsymbol{\mu}_c^m, \boldsymbol{\sigma}_c^m)$ to generate the kernel map $\mathbf{K}^m$.

Kernel Aggregation We use the normal distribution to generate an array of latent variable maps $\{\mathbf{Z}^m \mid m = 1, ..., N\}$. Each latent variable map can be used by Eq. (12) to produce a set of kernel maps $\{\mathbf{K}^m \mid m = 1, ..., N\}$. As illustrate in Figure 3(b), we sum these kernel maps as:

$$\mathbf{K}^u = \frac{1}{N} \sum_{m=1}^{N} \mathbf{K}^m, \tag{13}$$

where $\mathbf{K}^u \in \mathbb{R}^{H \times W \times (S \times S \times C)}$ is convoluted with the rainy image $\mathbf{I}$ for computing the object layer.

5 Experiments

5.1 Experimental Datasets

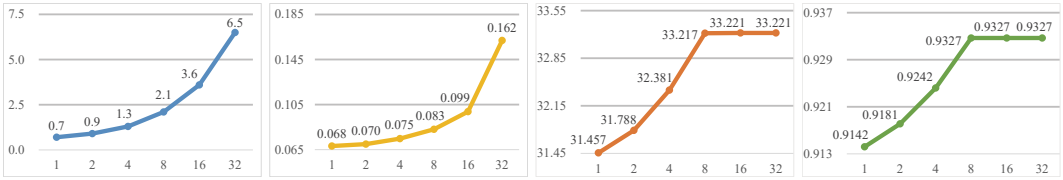
We compare SEIDNet with state-of-the-art methods, on the Rain100H [16], Rain100L [16], Rain1400 [19], and SPA [3] datasets. These datasets provide $1,800$/200/$12,600$/$638,492$ images for training, along with 100/100/$1,400$/$1,000$ images for testing. We also evaluate different methods on the Rain13K [10] dataset, which provides $13,712$ images for training. The test set of Rain13K contains $4,300$ images, which are taken from the test sets of Test100 [23], Test1200 [4], Test2800 [19], Rain100H, and Rain100L. Rain13K allows the models to be trained on the unified data and evaluated on the separate test sets, thus justifying the model generalization. We report the performances of rain removal in terms of peak signal-to-noise ratio (PSNR) and structural similarity (SSIM).

5.2 Ablation Study of SEIDNet

Below, we use the test set of Rain100H [16] for the major evaluation of SEIDNet.

Sensitivity to the Number of Kernels Given the testing architecture of SEIDNet, we generate and aggregate multiple kernels for removing the rains on the image. We change the number N of the generated kernels for each pixel, evaluating the impact on the performances. We choose the number N from the set $\{1, 2, 4, 8, 16, 32\}$. We report the computational overheads (i.e., GPU memory and testing time) in Figure 4(a–b), along with the performances (i.e., PSNR and SSIM) in Figure 4(c–d).

https://doi.org/10.52202/068431-0333



(a) GPU Memory (b) Testing Time (c) PSNR (d) SSIM

Figure 4: Sensitivities of GPU memory in GB (a), testing time per image in seconds (b), PSNR (c) and SSIM (d) to the number of kernels.

With $N = 1$, we generate a single kernel for each pixel on the rainy image, yielding 31.457 PSNR and 0.9142 SSIM on the test set of Rain100H. With $N = 8$, we considerably improve the performances (up to 33.217 PSNR and 0.9327 SSIM). This is because more kernels better capture the visual patterns of rain streaks and object textures. Too many kernels (e.g., $N = 32$) saturate the performances, but needing more computations. Below, we use $N = 8$ as default.

Analysis of Network Components The status estimation and information decoupling of SEIDNet are based on the generative CVAEs for computing the statuses and kernels. To evaluate the generative power for rain removal, we experiment with using the discriminative networks for the status estimation and information decoupling. For the status estimation, we resort to the convolution for predicting a single status map, based on the feature map of the rainy image. For the information decoupling, we pass the visual feature map and the status map to the convolution, predicting a single kernel map for the rainy image. The alternative architectures are provided in the supplementary material.

SE	ID	Memory	Time	PSNR	SSIM
D	**D**	**1.0**	**0.0515**	29.43	0.8809
G	**D**	1.2	0.0643	29.62	0.8849
D	**G**	1.9	0.0704	31.42	0.9172
G	**G**	2.1	0.0832	**33.22**	**0.9327**

Table 1: We refer to **SE**, **ID**, **Memory** and **Time** as the status estimation, information decoupling, GPU memory (GB) and testing time (second). **D** and **G** indicate the discriminative and generative networks, respectively. The performances are reported on the test set of Rain100H.

First, we remove all CVAEs, only using the discriminative network for the status estimation and information decoupling. In this case, we yield the unsatisfactory 29.43 PSNR and 0.8809 SSIM (see the first row of Table 1). The discriminative network only provides a pair of status and kernel maps for the rainy image, missing the critical patterns of rains and objects.

Next, we use the discriminative network for either the status estimation or the information decoupling. There is a CVAE for capturing the diverse patterns of the statuses or kernels, leading to better results (see the second and third rows of Table 1) than the discriminative networks. Yet, the performances achieved by a CVAE are lower than SEIDNet. This is because SEIDNet employs a couple of CVAEs to comprehensively model the correlation between each pair of status and kernel. We analyze the correlation between the status and the kernel in the supplementary material.

Various Combinations of Networks In Table 2, we compare different strategies of using the discriminative and generative networks for deraining. Similar to SEIDNet, we use 132 convolutional layers to construct a discriminative network for predicting the kernel maps. The generative SEIDNet outperforms the discriminative network (see the first and second rows).

Network	Memory	Time	PSNR	SSIM
D	**0.9**	**0.0453**	29.91	0.8905
G	2.1	0.0832	**33.22**	**0.9327**
D+G	2.4	0.0879	33.11	0.9310

Table 2: **D** or **G** means the discriminative or generative network that estimates the kernels for rain removal. The performances are reported on the test set of Rain100H.

We combine the discriminative and generative networks by averaging the kernel maps. This combination increases the network parameters but degrades the performances (see the last row). A smarter combination of the discriminative and generative networks is needed for reducing the parameters and improving the performances. We illustrate the compared networks in the supplementary material.

Different Ways of Using CVAEs SEIDNet has a pair of CVAEs that model the factorized distributions of the status and the kernel. We compare SEIDNet with the alternative methods, which use a CVAE without distribution factorization. We list the results in Table 3. Again, we illustrate the compared networks in the supplementary material.

Method	Memory	Time	PSNR	SSIM
One CVAE for **K**	**1.6**	**0.0703**	25.21	0.7929
One CVAE for (**K**, **R**)	1.9	0.0774	29.03	0.8963
Two CVAEs for (**K**, **R**)	2.1	0.0832	**33.22**	**0.9327**

Table 3: **K/R** is the kernel/status maps yielded by various probability factorizations. We list the results on the test set of Rain100H.

https://doi.org/10.52202/068431-0333

First, we set the single CVAE that only takes the visual feature map of the rainy image as the condition. This method directly generates the kernel maps. Because the status maps are unavailable for enabling the focused learning of the kernel maps, this method yields lower performances (see the first row).

Figure 5: Visual results on the image deraining task. We zoom in the image regions (see the blue rectangles) to compare deraining results of different methods.

Second, we use the single CVAE to generate the status and kernel maps. Again, this CVAE takes the visual feature map of the rainy image as the only condition. The status and kernel maps are generated by the separate decoder branches. The single CVAE only depends on the training loss of the status map, for implicitly guiding the generation of the kernel map. It is less effective than SEIDNet, where the generation of the kernel map is straightforwardly guided by the status map. As a result, SEIDNet outperforms this single CVAE (see the last two rows of Table 3).

Method	Subset A		Subset B		Subset C	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
MPR [10]	35.57	0.9519	36.83	0.9579	37.65	0.9630
EfDeRain [8]	34.37	0.9556	36.18	0.9629	35.44	0.9576
SPDNet [18]	31.84	0.9094	33.57	0.9254	32.93	0.9162
Our SEIDNet	**39.07**	**0.9813**	**39.90**	**0.9821**	**39.97**	**0.9799**

Table 4: We compare SEIDNet with state-of-the-art methods on three subsets of SPA dataset. The performances are reported in terms of PSNR and SSIM.

Method	Rain100H		Rain100L		Rain1400		SPA		Rain13K	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
PReNet [11]	29.46	0.8988	37.48	0.9792	32.66	0.9347	40.16	0.9816	31.47	0.9129
JORDER [16]	30.50	0.8967	38.59	0.9834	32.00	0.9347	40.78	0.9811	31.39	0.9118
SPANet [3]	25.11	0.8332	35.33	0.9694	29.85	0.9148	40.23	0.9838	29.83	0.8951
RCDNet [38]	31.28	0.9090	39.99	0.9860	33.04	0.9339	41.47	0.9854	32.02	0.9411
CVID [36]	27.93	0.8765	37.83	0.9882	28.69	0.8722	34.45	0.9437	28.19	0.8584
MPR [10]	30.64	0.9040	34.54	0.9564	33.28	0.9540	43.59	0.9879	33.27	0.9510
EfDeRain [8]	31.14	0.8990	35.04	0.9634	32.91	0.9323	43.77	0.9894	32.11	0.9416
SPDNet [18]	32.68	0.9202	39.59	0.9854	32.89	0.9444	43.55	0.9875	32.01	0.9367
Our SEIDNet	**33.22**	**0.9327**	**40.67**	**0.9865**	**34.84**	**0.9626**	**44.96**	**0.9911**	**33.62**	**0.9539**

Table 5: We compare SEIDNet with state-of-the-art methods on the test sets of Rain100H, Rain100L, Rain1400, SPA and Rain13K. The performances are reported in terms of PSNR and SSIM.

Discussion on Size of Training and Testing Dataset Conducting experiments on the unbalanced training and testing splits may reduce the confidence of SEIDNet. Thus, we reduce the training data in the extremely unbalanced dataset (i.e., SPA dataset with 638K/1K images for training and testing). This is done by randomly sampling 1K training images from SPA dataset. The random sample is three-fold, thus forming three different subsets, each of which contains 1K images for training different models. The trained models are evaluated on the 1K images in the original test set. With different subsets for training, SEIDNet yields better results than other methods (see the Table 4).

https://doi.org/10.52202/068431-0333

Method	Rain100H		Rain100L		Test100		Test1200		Test2800		Overall	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
PReNet [11]	27.02	0.8655	32.61	0.9513	24.89	0.8564	31.54	0.9136	31.79	0.9151	31.47	0.9130
JORDER [16]	27.43	0.8677	32.42	0.9476	24.29	0.8542	31.44	0.9110	31.72	0.9145	31.39	0.9118
SPANet [3]	26.88	0.8536	31.26	0.9247	23.17	0.7853	29.93	0.8928	30.07	0.9004	29.83	0.8951
RCDNet [38]	30.17	0.8876	35.06	0.9603	23.79	0.8303	31.68	0.9294	32.41	0.9513	32.02	0.9411
CVID [36]	26.25	0.8444	30.53	0.9025	23.23	0.7824	27.88	0.8401	28.50	0.8685	28.19	0.8584
MPR [10]	30.47	0.8926	36.45	0.9669	**30.29**	0.9139	32.98	0.9397	33.47	0.9587	33.26	0.9510
EfDeRain [8]	30.44	0.8954	35.45	0.9645	27.67	0.8874	31.41	0.9260	32.53	0.9511	32.11	0.9416
SPDNet [18]	30.56	0.8956	35.37	0.9621	24.87	0.8349	31.49	0.9152	32.59	0.9501	32.12	0.9367
Our SEIDNet	**31.18**	**0.8993**	**36.83**	**0.9657**	**30.29**	**0.9148**	**33.16**	**0.9442**	**33.93**	**0.9611**	**33.62**	**0.9539**

Table 6: We compare SEIDNet with state-of-the-art methods on the test sets of Rain13K. The performances are reported in terms of PSNR and SSIM.

Method	Snow100K-S		Snow100K-M		Snow100K-L		Overall	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
DuRN-S-P [44]	32.27	0.9497	30.92	0.9398	27.21	0.8891	30.12	0.9261
Composition GAN [45]	30.43	0.9612	31.21	0.9431	29.55	0.9021	30.40	0.9335
DesnowNet [20]	32.33	0.9500	30.87	0.9409	27.17	0.8983	30.11	0.9296
DS-GAN [46]	33.43	0.9641	31.88	0.9570	28.07	0.9211	31.11	0.9473
HDCWNet [47]	33.21	0.9623	32.38	0.9541	28.13	0.9253	31.24	0.9472
DDMSNet [48]	34.34	0.9445	32.89	0.9330	28.85	0.8772	32.03	0.9183
RFMPRaLSGAN [49]	33.68	0.9690	30.47	0.9500	29.38	0.9440	31.17	0.9540
RSRNet [50]	31.54	0.9519	30.52	0.9444	26.85	0.9039	29.64	0.9334
Our SEIDNet	**35.01**	**0.9765**	**33.45**	**0.9711**	**29.84**	**0.9454**	**32.77**	**0.9643**

Table 7: We compare SEIDNet with state-of-the-art methods on the test sets of Snow100K. The performances are reported in terms of PSNR and SSIM.

5.3 Comparison with State-of-the-Art Methods

In Table 5, we compare SEIDNet with the recent methods on the Rain100H, Rain100L, Rain1400, SPA, and Rain13K datasets, where SEIDNet outperforms other methods. We show the deraining results of the competitive methods in Figure 5.

SEIDNet is trained on the unified data, outperforming other methods on 5 test sets of Rain13K. Thus, SEIDNet shows a better generalization across different test sets. We average the performances on 5 test sets and report the results in Table 5. We provide the results on the separate test sets (i.e., the test sets of Test100, Test1200 [4], Test2800 [19], Rain100H, and Rain100L) in the Table 6. The proposed SEIDNet outperforms ohter methods on the separate test sets, which justifies the generalization of the deraining model.

Method	ITS Subset		OTS Subset	
	PSNR	SSIM	PSNR	SSIM
Grid-Net [39]	32.16	0.9836	30.86	0.9820
MSBDN [40]	33.67	0.9850	33.48	0.9820
FFA-Net [41]	36.39	0.9886	33.57	0.9840
AECR-Net [42]	37.17	0.9901	33.84	0.9837
D-Former [43]	40.05	0.9960	34.95	0.9840
Our SEIDNet	**40.62**	**0.9968**	**35.72**	**0.9951**

Table 8: We compare SEIDNet with other methods on ITS&OTS. The results are listed in terms of PSNR and SSIM.

5.4 Extensive Comparison on Different Tasks

To evaluate the generalization of SEIDNet on differen tasks, we use SEIDNet to resolve the desnow, dehaze, and deshadow tasks. We report the performances on different tasks in Tables 7, 8, 9, and 10. We provide the visual results of SEIDNet on different tasks in Figure 6.

In Table 7, we use Snow100K dataset [20] for evaluation. Snow100K has three subsets, i.e., Snow100K-S, Snow100K-M, and Snow100K-L, where the snow flakes in the images have small, medium, and large sizes. The images of Snow100K-S only contains the small snow flakes. The snow flakes in Snow100K-M have small and medium sizes, while those in Snow100K-L have small, medium, and large sizes. In each subset, there are about 17K images for training/testing. We also average the results (see **"Overall"**) on Snow100K-S, Snow100K-M, and Snow100K-L.

Method	Shadow	Non-Shadow	All
ST-CGAN [22]	13.4	7.7	8.7
DeshadowNet [51]	15.9	6.0	7.6
Mask-GAN [52]	12.4	4.0	5.3
SP+M-Net [53]	9.7	3.0	4.0
PMDNet [54]	9.7	3.0	4.0
AEFNet [55]	6.5	3.8	4.2
CRFormer [56]	**5.9**	**2.9**	**3.4**
Our SEIDNet	6.4	3.4	3.9

Table 9: We compare SEIDNet with other methods on the test set of ISTD+. The performances are reported in term of RMSE.

In Table 8, we use the ITS&OTS dataset [21] to justify the generalization of SEIDNet on the haze removal task. In the ITS&OTS dataset, the ITS subset

https://doi.org/10.52202/068431-0333

Figure 6: Visual results on Snow100K, ITS&OTS and ISTD test sets.

contains 110,000 indoor images with haze. The OTS subset has 313,950 outdoor images with haze. We train SEIDNet on the ITS and OTS subsets, respectively. The network is evaluated on the indoor and outdoor test sets, respectively, where each test set contains 500 images. In Table 8, we also report the results of other methods for haze removal.

In Table 9 and 10, we use the adjusted ISTD (ISTD+) [57] and ISTD [22] datasets to evaluate the performances of different methods on the shadow removal task. The images in the ISTD dataset are taken from 135 different scenarios. There are 1,330 and 540 images for training and testing, respectively. Le et al. [57] adjust the color inconsistency between the shadow and shadow free images of ISTD and achieve the ISTD+ dataset. We use the image regions with/without shadow to test different methods (see the performances in **"Shadow"** and **"Non-Shadow"**). We also use the full images to test the methods and report the performances in **"All"**.

6 Conclusions

The latest progress of rain removal benefits from deep discriminative networks trained on large-scale datasets. In this paper, we have proposed a generative network, SEIDNet, to generate the pixel-wise status and kernel for rain removal. SEIDNet has two CVAEs, which model the factorized probability distributions. It learns the status and kernel spaces. In contrast to the discriminative networks, SEIDNet enables the generation of multiple statuses for the pixel, for capturing the confusion between the appearances of rains and objects. We employ these statuses for generating multiple kernels, reducing the confusing information and refining the deraining result on the pixel. SEIDNet achieves state-of-the-art performances on the public datasets. In the future work, we plan to explore an effective strategy of combining the discriminative and generative networks for rain removal.

Method	Shadow	Non-Shadow	All
Mask-GAN [52]	12.67	6.68	7.41
ARGAN [58]	9.21	6.27	6.63
DSC [59]	9.22	6.39	6.67
RIS-GAN [60]	9.15	6.31	6.62
DHAN [61]	8.14	6.04	6.37
CANet [62]	8.86	6.07	6.15
AEFNet [55]	7.77	5.56	5.92
CRFormer [56]	**7.32**	5.82	6.07
Our SEIDNet	7.47	**5.08**	**5.47**

Table 10: We compare SEIDNet with other methods on the test set of ISTD. The performances are reported in term of RMSE.

Negative Societal Impacts

Our approach can be broadly applied in many scenarios (e.g., autonomous vehicles and video surveillance). One should be cautious of the problematic results, which may give rise to the infringement of privacy or economic interest.

Acknowledgements

We thank the anonymous reviewers for their constructive comments. This work was supported by The Hong Kong Polytechnic University under Grant P0030419, Grant P0042740, and Grant P0035358.

https://doi.org/10.52202/068431-0333

References

[1] Xiaowei Hu, Chiwing Fu, Lei Zhu, and Phengann Heng. Depth-attentional features for single-image rain removal. In *IEEE CVPR*, pages 8022–8031, 2019.

[2] Hong Wang, Qi Xie, Qian Zhao, and Deyu Meng. A model-driven deep neural network for single image rain removal. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3103–3112, 2020.

[3] Tianyu Wang, Xin Yang, Ke Xu, Shaozhe Chen, Qiang Zhang, and Rynson W.H. Lau. Spatial attentive single-image deraining with a high quality real rain dataset. In *IEEE CVPR*, pages 12270–12279, 2019.

[4] He Zhang and Vishal M Patel. Density-aware single image de-raining using a multi-stream dense network. In *IEEE CVPR*, pages 695–704, 2018.

[5] Yang Liu, Ziyu Yue, Jinshan Pan, and Zhixun Su. Unpaired learning for deep image deraining with rain direction regularizer. In *IEEE ICCV*, pages 4753–4761, 2021.

[6] Ke Xu, Xin Tian, Xin Yang, Baocai Yin, and Rynson WH Lau. Intensity-aware single-image deraining with semantic and color regularization. *IEEE Transactions on Image Processing*, 30:8497–8509, 2021.

[7] Yinglong Wang, Chao Ma, and Bing Zeng. Multi-decoding deraining network and quasi-sparsity based training. In *IEEE CVPR*, pages 13375–13384, 2021.

[8] Qing Guo, Jingyang Sun, Felix Juefei-Xu, Lei Ma, Xiaofei Xie, Wei Feng, Yang Liu, and Jianjun Zhao. Efficientderain: Learning pixel-wise dilation filtering for high-efficiency single-image deraining. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1487–1495, 2021.

[9] Ye-Tao Wang, Xi-Le Zhao, Tai-Xiang Jiang, Liang-Jian Deng, Yi Chang, and Ting-Zhu Huang. Rain streaks removal for single image via kernel-guided convolutional neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3664–3676, 2020.

[10] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *IEEE CVPR*, pages 14821–14831, 2021.

[11] Dongwei Ren, Wangmeng Zuo, Qinghua Hu, Pengfei Zhu, and Deyu Meng. Progressive image deraining networks: A better and simpler baseline. In *IEEE CVPR*, pages 3937–3946, 2019.

[12] Xia Li, Jianlong Wu, Zhouchen Lin, Hong Liu, and Hongbin Zha. Recurrent squeeze-and-excitation context aggregation net for single image deraining. In *ECCV*, pages 254–269, 2018.

[13] Ruoteng Li, Loong-Fah Cheong, and Robby T Tan. Single image deraining using scale-aware multi-stage recurrent network. *arXiv preprint arXiv:1712.06830*, 2017.

[14] Ruoteng Li, Loong-Fah Cheong, and Robby T Tan. Heavy rain image restoration: Integrating physics model and conditional adversarial learning. In *IEEE CVPR*, pages 1633–1642, 2019.

[15] Wenhan Yang, Robby T Tan, Jiashi Feng, Jiaying Liu, Zongming Guo, and Shuicheng Yan. Deep joint rain detection and removal from a single image. In *IEEE CVPR*, pages 1357–1366, 2017.

[16] Wenhan Yang, Robby T Tan, Jiashi Feng, Jiaying Liu, Shuicheng Yan, and Zongming Guo. Joint rain detection and removal from a single image with contextualized deep networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(6):1377–1393, 2019.

[17] Rui Qian, Robby T Tan, Wenhan Yang, Jiajun Su, and Jiaying Liu. Attentive generative adversarial network for raindrop removal from a single image. In *IEEE CVPR*, pages 2482–2491, 2018.

[18] Qiaosi Yi, Juncheng Li, Qinyan Dai, Faming Fang, Guixu Zhang, and Tieyong Zeng. Structure-preserving deraining with residue channel prior guidance. In *IEEE ICCV*, pages 4238–4247, 2021.

[19] Xueyang Fu, Jiabin Huang, Delu Zeng, Huang Yue, and John Paisley. Removing rain from single images via a deep detail network. In *IEEE CVPR*, pages 3855–3863, 2017.

[20] Yun-Fu Liu, Da-Wei Jaw, Shih-Chia Huang, and Jenq-Neng Hwang. Desnownet: Context-aware deep network for snow removal. *IEEE Transactions on Image Processing*, 27(6):3064–3073, 2018.

[21] Boyi Li, Wenqi Ren, Dengpan Fu, Dacheng Tao, Dan Feng, Wenjun Zeng, and Zhangyang Wang. Benchmarking single-image dehazing and beyond. *IEEE Transactions on Image Processing*, 28(1):492–505, 2018.

[22] Jifeng Wang, Xiang Li, and Jian Yang. Stacked conditional generative adversarial networks for jointly learning shadow detection and shadow removal. In *IEEE CVPR*, pages 1788–1797, 2018.

[23] He Zhang, Vishwanath Sindagi, and Vishal M Patel. Image de-raining using a conditional generative adversarial network. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(11):3943–3956, 2019.

[24] Yu Luo, Yong Xu, and Hui Ji. Removing rain from a single image via discriminative sparse coding. In *IEEE ICCV*, pages 3397–3405, 2015.

[25] Yu Li, Robby T Tan, Xiaojie Guo, Jiangbo Lu, and Michael S Brown. Rain streak removal using layer priors. In *IEEE CVPR*, pages 2736–2744, 2016.

[26] Sen Deng, Mingqiang Wei, Jun Wang, Yidan Feng, Luming Liang, Haoran Xie, Fu Lee Wang, and Meng Wang. Detail-recovery image deraining via context aggregation networks. In *IEEE CVPR*, pages 14560–14569, 2020.

[27] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[28] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

[29] Zhijie Wu, Xiang Wang, Di Lin, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Sagnet: Structure-aware generative network for 3d-shape modeling. *ACM Transactions on Graphics*, 38(4):1–14, 2019.

[30] Xin Jin, Zhibo Chen, Jianxin Lin, Zhikai Chen, and Wei Zhou. Unsupervised single image deraining with self-supervised constraints. In *IEEE International Conference on Image Processing*, pages 2761–2765, 2019.

[31] Ian Goodfellow, Jean Pougetabadie, Mehdi Mirza, Bing Xu, David Wardefarley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, volume 27, 2014.

[32] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[33] Yanyan Wei, Zhao Zhang, Yang Wang, Mingliang Xu, Yi Yang, Shuicheng Yan, and Meng Wang. Deraincyclegan: Rain attentive cyclegan for single image deraining and rainmaking. *IEEE Transactions on Image Processing*, 30:4788–4801, 2021.

[34] Yuntong Ye, Yi Chang, Hanyu Zhou, and Luxin Yan. Closing the loop: Joint rain generation and removal via disentangled image translation. In *IEEE CVPR*, pages 2053–2062, 2021.

[35] Zheng Dong, Ke Xu, Yin Yang, Hujun Bao, Weiwei Xu, and Rynson WH Lau. Location-aware single image reflection removal. In *IEEE ICCV*, pages 5017–5026, 2021.

[36] Yingjun Du, Jun Xu, Xiantong Zhen, Ming-Ming Cheng, and Ling Shao. Conditional variational image deraining. *IEEE Transactions on Image Processing*, 29:6288–6301, 2020.

[37] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *NeurIPS*, pages 3483–3491, 2015.

[38] Hong Wang, Qi Xie, Qian Zhao, and Deyu Meng. A model-driven deep neural network for single image rain removal. In *IEEE CVPR*, pages 3103–3112, 2020.

[39] Xiaohong Liu, Yongrui Ma, Zhihao Shi, and Jun Chen. Griddehazenet: Attention-based multi-scale network for image dehazing. In *IEEE ICCV*, pages 7314–7323, 2019.

[40] Hang Dong, Jinshan Pan, Lei Xiang, Zhe Hu, Xinyi Zhang, Fei Wang, and Ming-Hsuan Yang. Multi-scale boosted dehazing network with dense feature fusion. In *IEEE CVPR*, pages 2157–2167, 2020.

[41] Xu Qin, Zhilin Wang, Yuanchao Bai, Xiaodong Xie, and Huizhu Jia. Ffa-net: Feature fusion attention network for single image dehazing. In *AAAI*, pages 11908–11915, 2020.

[42] Haiyan Wu, Yanyun Qu, Shaohui Lin, Jian Zhou, Ruizhi Qiao, Zhizhong Zhang, Yuan Xie, and Lizhuang Ma. Contrastive learning for compact single image dehazing. In *IEEE CVPR*, pages 10551–10560, 2021.

[43] Yuda Song, Zhuqing He, Hui Qian, and Xin Du. Vision transformers for single image dehazing. *arXiv preprint arXiv:2204.03883*, 2022.

https://doi.org/10.52202/068431-0333

[44] Xing Liu, Masanori Suganuma, Zhun Sun, and Takayuki Okatani. Dual residual networks leveraging the potential of paired operations for image restoration. In *IEEE CVPR*, pages 7007–7016, 2019.

[45] Zhi Li, Juan Zhang, Zhijun Fang, Bo Huang, Xiaoyan Jiang, Yongbin Gao, and Jenq-Neng Hwang. Single image snow removal via composition generative adversarial networks. *IEEE Access*, 7:25016–25025, 2019.

[46] Da-Wei Jaw, Shih-Chia Huang, and Sy-Yen Kuo. Desnowgan: An efficient single image snow removal framework using cross-resolution lateral connection and gans. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(4):1342–1350, 2020.

[47] Wei-Ting Chen, Hao-Yu Fang, Cheng-Lin Hsieh, Cheng-Che Tsai, I Chen, Jian-Jiun Ding, Sy-Yen Kuo, et al. All snow removed: Single image desnowing algorithm using hierarchical dual-tree complex wavelet representation and contradict channel loss. In *IEEE ICCV*, pages 4196–4205, 2021.

[48] Kaihao Zhang, Rongqing Li, Yanjiang Yu, Wenhan Luo, and Changsheng Li. Deep dense multi-scale network for snow removal using semantic and depth priors. *IEEE Transactions on Image Processing*, 30:7419–7431, 2021.

[49] Thaileang Sung and Hyo Jong Lee. Removing snow from a single image using a residual frequency module and perceptual ralsgan. *IEEE Access*, 9:152047–152056, 2021.

[50] Hamidreza Fazlali, Shahram Shirani, Michael Bradford, and Thia Kirubarajan. Single image rain/snow removal using distortion type information. *Multimedia Tools and Applications*, pages 1–27, 2022.

[51] Liangqiong Qu, Jiandong Tian, Shengfeng He, Yandong Tang, and Rynson WH Lau. Deshadownet: A multi-context embedding deep network for shadow removal. In *IEEE CVPR*, pages 4067–4075, 2017.

[52] Xiaowei Hu, Yitong Jiang, Chi-Wing Fu, and Pheng-Ann Heng. Mask-shadowgan: Learning to remove shadows from unpaired data. In *IEEE ICCV*, pages 2472–2481, 2019.

[53] Hieu Le and Dimitris Samaras. Physics-based shadow image decomposition for shadow removal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (01):1–1, 2021.

[54] Hieu Le and Dimitris Samaras. From shadow segmentation to shadow removal. In *ECCV*, pages 264–281, 2020.

[55] Lan Fu, Changqing Zhou, Qing Guo, Felix Juefei-Xu, Hongkai Yu, Wei Feng, Yang Liu, and Song Wang. Auto-exposure fusion for single-image shadow removal. In *IEEE CVPR*, pages 10571–10580, 2021.

[56] Jin Wan, Hui Yin, Zhenyao Wu, Xinyi Wu, Zhihao Liu, and Song Wang. Crformer: A cross-region transformer for shadow removal. *arXiv preprint arXiv:2207.01600*, 2022.

[57] Hieu Le and Dimitris Samaras. Shadow removal via shadow image decomposition. In *IEEE ICCV*, pages 8578–8587, 2019.

[58] Bin Ding, Chengjiang Long, Ling Zhang, and Chunxia Xiao. Argan: Attentive recurrent generative adversarial network for shadow detection and removal. In *IEEE ICCV*, pages 10213–10222, 2019.

[59] Xiaowei Hu, Chi-Wing Fu, Lei Zhu, Jing Qin, and Pheng-Ann Heng. Direction-aware spatial context features for shadow detection and removal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(11):2795–2808, 2019.

[60] Ling Zhang, Chengjiang Long, Xiaolong Zhang, and Chunxia Xiao. Ris-gan: Explore residual and illumination with generative adversarial networks for shadow removal. In *AAAI*, pages 12829–12836, 2020.

[61] Xiaodong Cun, Chi-Man Pun, and Cheng Shi. Towards ghost-free shadow removal via dual hierarchical aggregation network and shadow matting gan. In *AAAI*, pages 10680–10687, 2020.

[62] Zipei Chen, Chengjiang Long, Ling Zhang, and Chunxia Xiao. Canet: A context-aware network for shadow removal. In *IEEE ICCV*, pages 4743–4752, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please see the section of **Limitation** in the supplementary material.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please see the section of **Negative Societal Impacts**.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Please see the section of **Implementation Details** in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please see the section of **Implementation Details** in the supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see the section of **Implementation Details** in the supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

https://doi.org/10.52202/068431-0333