Generative Status Estimation and Information Decoupling for Image Rain Removal – Supplementary Material –

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

1 Experimental Details

We construct SEIDNet based on PyTorch[1]. There are 26 convolutional layers for extracting the visual feature map from the rainy image. The feature masking contains two convolutional layers. It computes the rain (or object) feature map. The encoder/condition/decoder branch of CVAE ($\mathcal{V}_{se}$ or $\mathcal{V}_{id}$) consists of 16 convolutional layers. There is a pair of batch normalization and ReLU layers between the adjacent convolutional layers. The size of kernels in each convolutional layer is 3×3. $\mathcal{V}_{id}$ generates 3×3 kernel for deraining each pixel. There are 132 convolutional layers in SEIDNet.

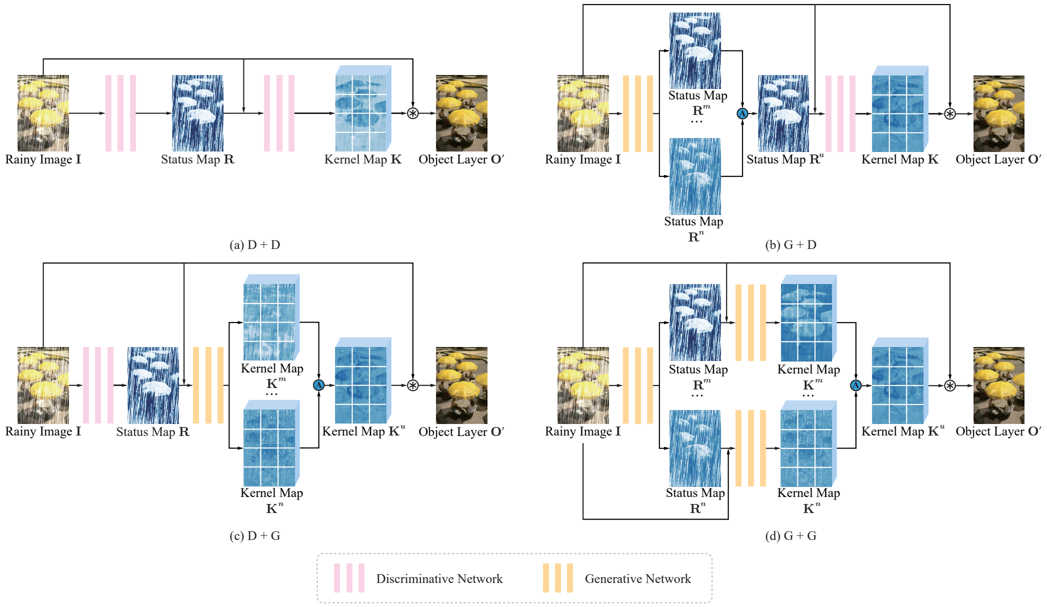


Figure 1: Different strategies of using the status estimation and information decoupling.

We use Adam solver to optimize the parameters of SEIDNet. We set the initial learning rate to $5e-5$, which is decayed linearly. We train the network for $5,000$ epoches on the dataset, where we randomly

[1] https://pytorch.org/

select 10 images for constructing each mini-batch. The spatial resolution of the image is 256 × 256. We use a GeForce RTX 3090 GPUs for training. Given the testing architecture of SEIDNet, we generate 8 kernel maps, which are aggregated for the deraining on the rainy image.

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.

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: **Network** means the discriminative (abbreviated as **D**) or generative (abbreviated as **G**) network that estimates the kernels for rain removal. The performances are reported on the test set of Rain100H.

2 Ablation Study of SEIDNet

Analysis of Network Components The status estimation and information decoupling of SEIDNet are the core components for computing the pixel-wise statuses and kernels. These components are based on the generative CVAEs. To evaluate the generative power for rain removal, we experiment with using the discriminative convolutional network for the status estimation and information decoupling. We report the results in Table 1 (also see Table 1 of the main paper). In this supplementary material, we illustrate the architectures in Figure 1(a–d).

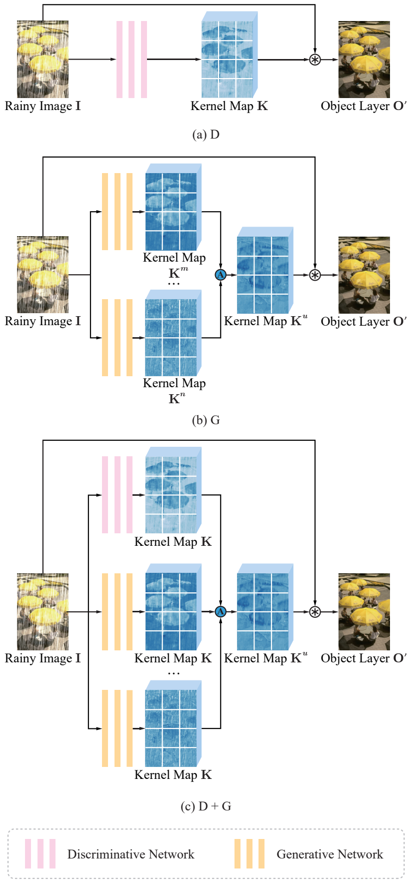
Various Combinations of Networks In Table 2 (also see Table 2 of the main paper), we compare different strategies of using the discriminative and generative networks for deraining. Similar to the full model of SEIDNet (see Figure 2(a)), we use 132 convolutional layers to construct a single discriminative network for predicting the pixel-wise kernels. The architecture of the discriminative network is illustrated in Figure 2(b).

We combine the discriminative and generative networks, by averaging the kernel maps produced by the discriminative and generative networks. The combined networks are illustrated in Figure 2(c). The trivial combination increases the network parameters but degrades the performances.

Different Ways of Using CVAEs SEIDNet has a pair of CVAEs that model the factorized distributions of the status and the kernel (see Figure 3(c)). We compare SEIDNet with the alternative methods, which use a CVAE without distribution factorization. We list the results in Table 3 (also see Table 3 of the main paper).

First, we experiment with the single CVAE that only takes the visual feature map of the rainy image as the condition. This method directly generates the kernel maps (see Figure 3(a)).

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.


Rainy Image $\mathbf{I}$
Kernel Map $\mathbf{K}$
Object Layer $\mathbf{O}'$
Kernel Map $\mathbf{K}^m$
Kernel Map $\mathbf{K}^n$
Kernel Map $\mathbf{K}^u$
A
Discriminative Network
Generative Network

(a) D

(b) G

(c) D + G

Figure 2: Various network combinations.

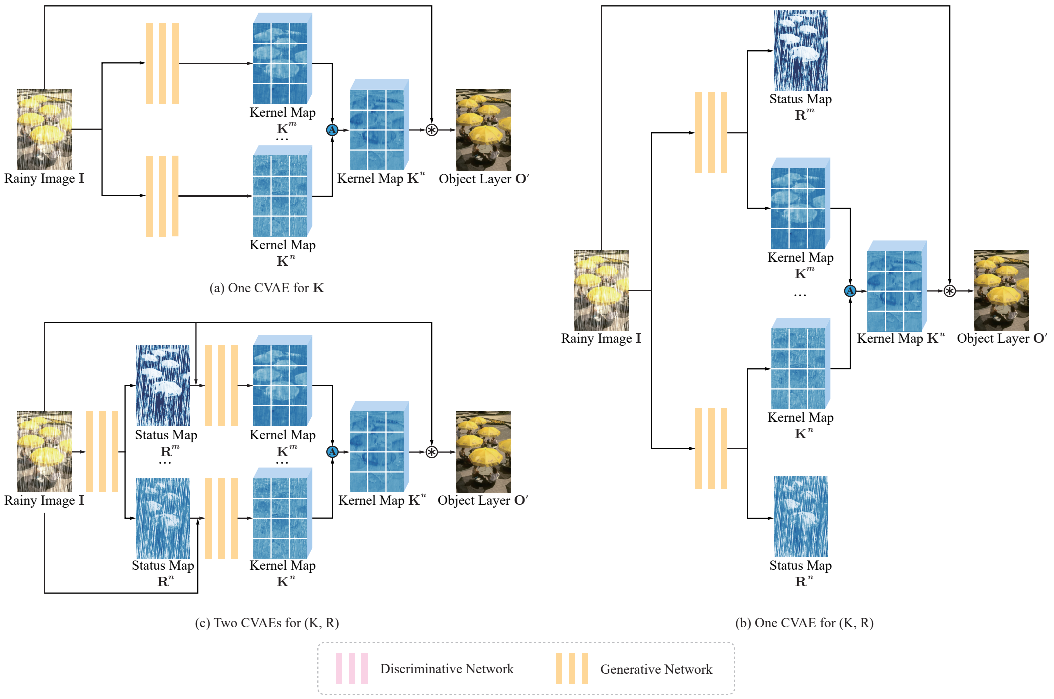
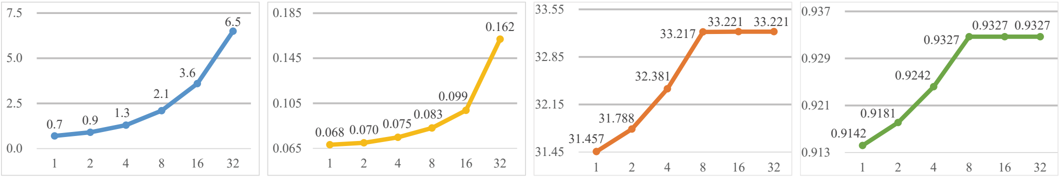


Figure 3: We show the simplified network structure of different ways of using CVAEs.

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

Table 3: We refer to $\mathbf{K}$ and $\mathbf{R}$ as the kernel and status maps. We use different factorizations of probability and report the performances on the test set of Rain100H.

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 (see Figure 3(b)).



(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.

Visualization of Status Maps We use the normal distribution to generate an array of latent variable maps $\{\mathbf{Z}^m \mid m = 1, ..., N\}$. The decoder of $\mathcal{V}_{se}$ uses feature map $\mathbf{F}$, latent variable maps $\mathbf{Z}^m$ and $(\boldsymbol{\mu}_f, \boldsymbol{\sigma}_f)$ to generate the status map $\{\mathbf{R}^m \mid m = 1, ..., N\}$. In Figure 6, we compare the visualized status maps, which are estimated from the testing images.

In each status map, a higher (or lower) value means a higher (or lower) probability of predicting the pixel as the rain (or object). For visualization, the pixel in blue (or white) has a higher (or lower) value. In each row of Figure 6, we zoom in some of the regions on the status maps (see the regions in the red rectangles) that are estimated from the identical image. These regions contain different status values for capturing the confusing appearance of rain and object.

Visualization of Kernel Maps The decoder of $\mathcal{V}_{id}$ uses $\mathbf{F}$, $\mathbf{R}^m$, $\mathbf{Z}^m$ and $(\boldsymbol{\mu}_c^m, \boldsymbol{\sigma}_c^m)$ to generate the kernel map $\{\mathbf{K}^m \mid m = 1, ..., N\}$. Given various status maps generated by $\mathcal{V}_{se}$, the decoder of $\mathcal{V}_{id}$

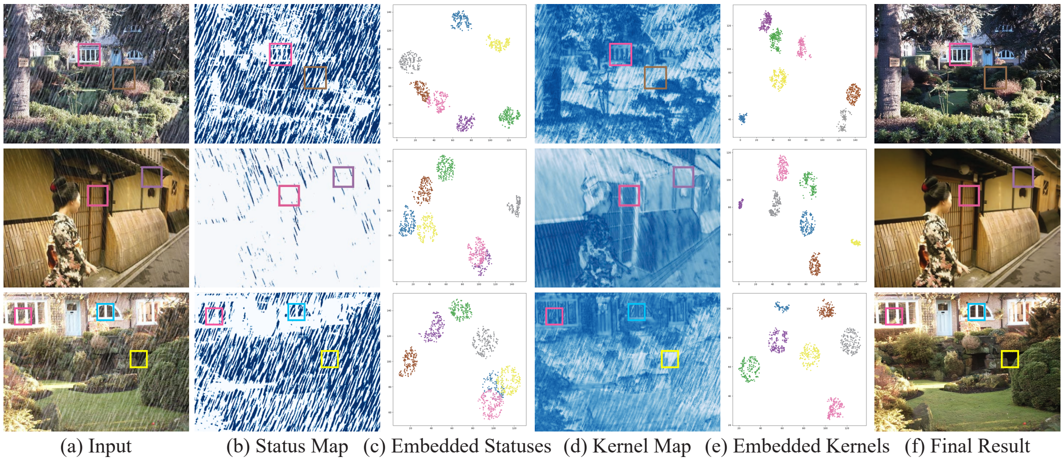


Figure 5: (a) We choose the chosen regions in each input image. (b) We show a status map for each input image. (c) For each chosen region, we use t-SNE to embed different counterparts of statuses into the 2D space. Here, each scatter point corresponds to a counterpart of statuses in a chosen region with the same color. (d) We show a kernel map for each input image. (e) For each chosen region, we use t-SNE to embed different counterparts of kernels into the 2D space. Here, each scatter point corresponds to a counterpart of kernels in a chosen region with the same color. (f) The final results achieved by SEIDNet.

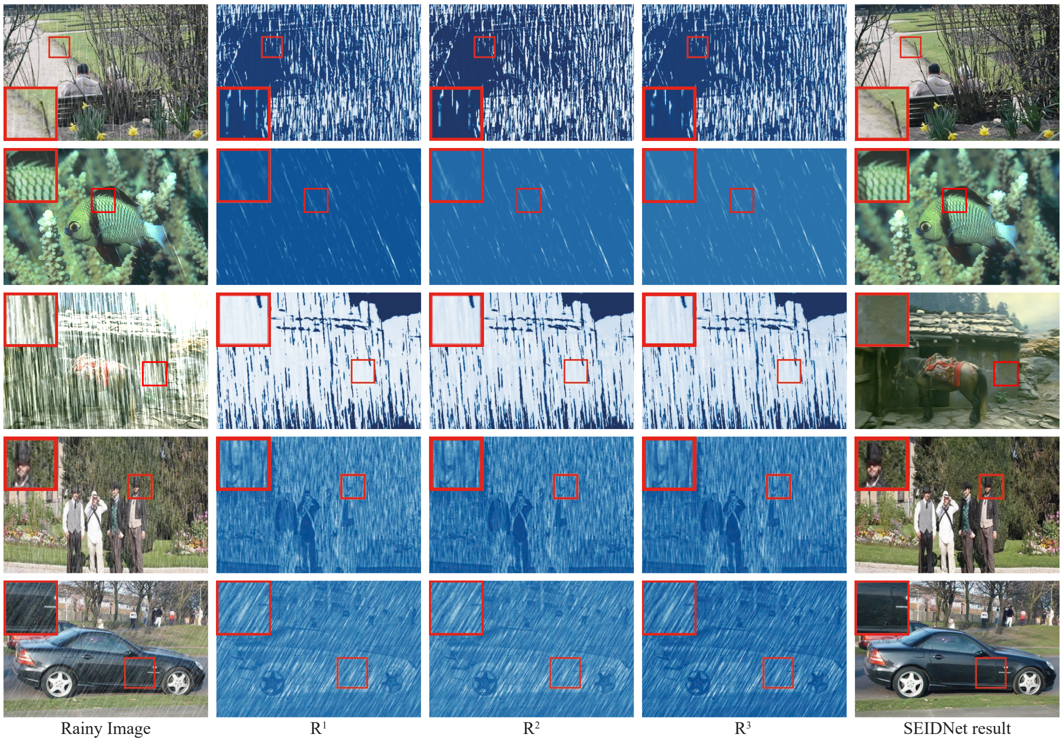


Figure 6: Visualization of status maps.

can generate various kernel maps for the same image. In Figure 7, we compare the visualized kernel maps $\mathbf{K}^m$, which are estimated from the testing images.

In each row of Figure 7, the sample regions on the kernel maps (see the red rectangles) are estimated from the identical image. Note that these regions correspond to different kernels for deraining. As evidenced in Figure 4 (also see Figure 4 of the main paper), increasing the number of the estimated kernel maps helps to improve the deraining performance.

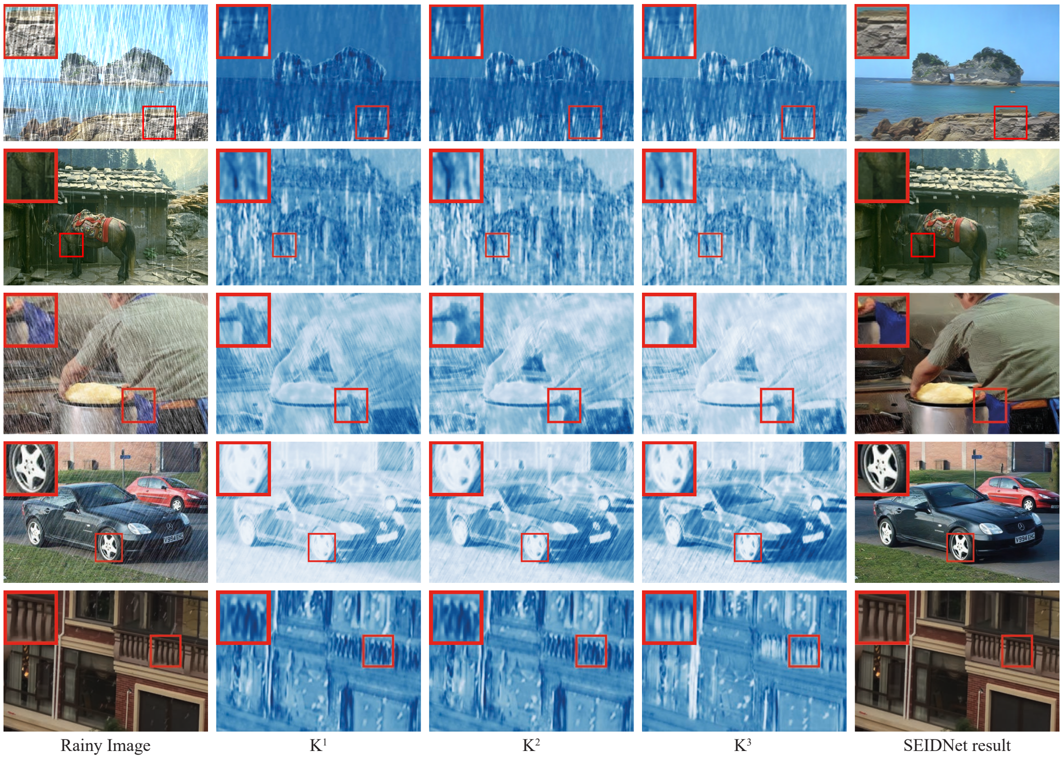


Figure 7: Visualization of kernel maps.

Method	Rain100H		Rain100L		Test100		Test1200		Test2800		Overall	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
PReNet [2]	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 [3]	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 [4]	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 [5]	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 [6]	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 [7]	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 [9]	30.56	0.8956	35.37	0.9621	24.87	0.8349	31.49	0.9152	32.59	0.9501	32.12	0.9367
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 4: 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.

Analysis on the Correlation between Statuses and Kernels To analyze the correlation between the status and kernel maps produced by SEIDNet, we employ t-SNE [1] to embed the status and kernel maps into the 2D space for visual analysis.

As shown in Figure 5(a), we choose the regions from the input images. In the chosen regions of each input image, the statuses of rains and objects are very similar (see Figure 5(b)). Thus, for the chosen region of each image, their different counterparts of statuses, which are sampled from the status space, are embedded closely into the 2D space (see Figure 5(c)). Intuitively, the overlapping of statuses in the 2D space likely let deraining kernels for the chosen regions be similar, thus leading to the failure in the deraining on these regions.

Note that the status estimation and information decoupling of SEIDNet are powerful. Though the statuses of the chosen regions are similar, multiple counterparts of status maps still provide the useful information, which helps the information decoupling to estimate the reliable kennels for different regions (see the kernel map in Figure 5(d)). As shown in Figure 5(e), we embed different counterparts of kernels of each chosen region into the 2D space, where the kernels of different regions are relatively far from each other. It helps to yield the satisfactory results in Figure 5(f).

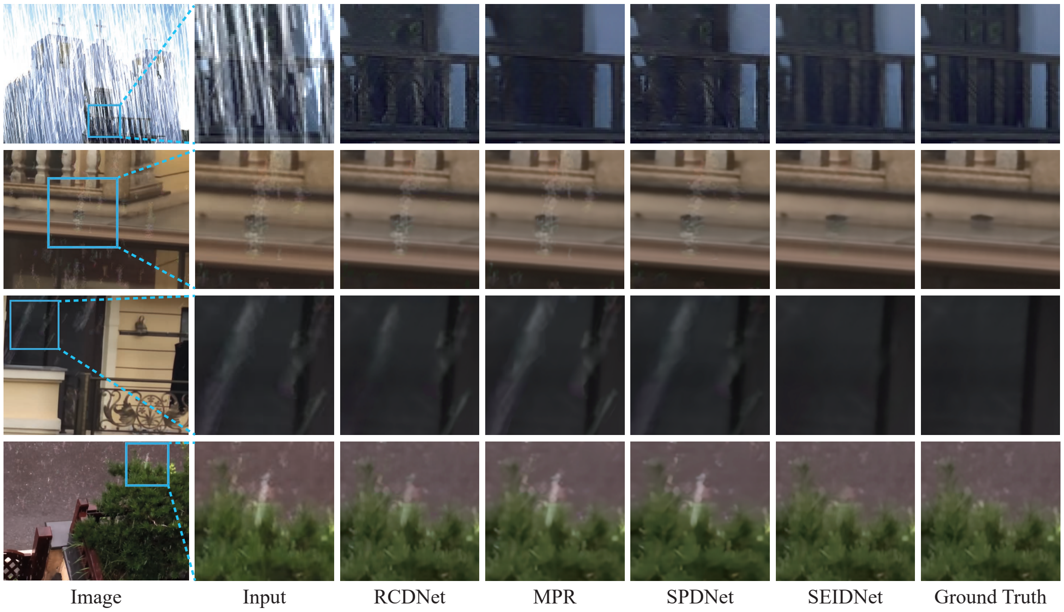


Figure 8: Visual results on the image deraining task. We zoom in the image regions (see the blue rectangles) to compare deraining performance of different methods.

Method	Snow100K-S		Snow100K-M		Snow100K-L		Overall	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
DuRN-S-P [12]	32.27	0.9497	30.92	0.9398	27.21	0.8891	30.12	0.9261
Composition GAN [13]	30.43	0.9612	31.21	0.9431	29.55	0.9021	30.40	0.9335
DesnowNet [14]	32.33	0.9500	30.87	0.9409	27.17	0.8983	30.11	0.9296
DS-GAN [15]	33.43	0.9641	31.88	0.9570	28.07	0.9211	31.11	0.9473
HDCWNet [16]	33.21	0.9623	32.38	0.9541	28.13	0.9253	31.24	0.9472
DDMSNet [17]	34.34	0.9445	32.89	0.9330	28.85	0.8772	32.03	0.9183
RFMPRaLSGAN [18]	33.68	0.9690	30.47	0.9500	29.38	0.9440	31.17	0.9540
RSRNet [19]	31.54	0.9519	30.52	0.9444	26.85	0.9039	29.64	0.9334
SEIDNet	**35.01**	**0.9765**	**33.45**	**0.9711**	**29.84**	**0.9454**	**32.77**	**0.9643**

Table 5: 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.

Method	ITS Subset		OTS Subset	
	PSNR	SSIM	PSNR	SSIM
Grid-Net [20]	32.16	0.9836	30.86	0.9820
MSBDN [21]	33.67	0.9850	33.48	0.9820
FFA-Net [22]	36.39	0.9886	33.57	0.9840
AECR-Net [23]	37.17	0.9901	33.84	0.9837
D-Former [24]	40.05	0.9960	34.95	0.9840
SEIDNet	**40.62**	**0.9968**	**35.72**	**0.9951**

Table 6: We compare SEIDNet with other methods on ITS&OTS. The results are listed in terms of PSNR and SSIM.

3 Supplementary Results on Rain Removal

In Table 4, we report the performances on the separate test sets (i.e., the test sets of Test100, Test1200 [10], Test2800 [11], Rain100H, and Rain100L) of Rain13K. We also compare the performances of different methods. Each method is trained on the unified training set of Rain13K and tested on the separate test sets. Again, SEIDNet outperforms other methods, showing a strong generalization across different datasets.

In Figures 8 and 9, we provide more visual results on the task of rain removal.

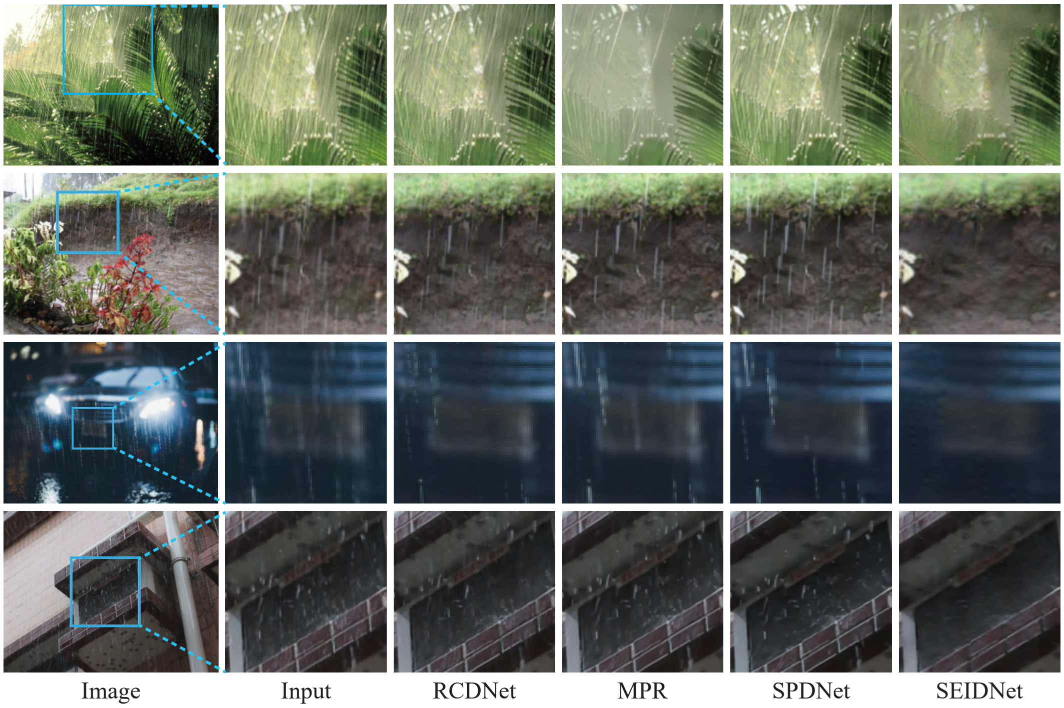


Figure 9: Visual results on real-world dataset. We zoom in the image regions (see the blue rectangles) to compare deraining performance of different methods.

4 Extensive Comparison on Different Tasks

Note that SEIDNet can be extended to various image restoration tasks. To evaluate the generalization of SEIDNet on differen tasks, we experiment with using SEIDNet to address the removal of snow, haze, and shadow. We report the performances of different methods in Tables 5, 6, 7, and 8 (also see Tables 7–10 of the main paper). In Figures 14, 15, and 16, we compare the visual results of different methods on the Snow100K, ITS&OTS, ISTD, and ISTD+ datasets.

Method	Shadow	Non-Shadow	All
Mask-GAN [25]	12.67	6.68	7.41
ARGAN [26]	9.21	6.27	6.63
DSC [27]	9.22	6.39	6.67
RIS-GAN [28]	9.15	6.31	6.62
DADNet [29]	8.65	6.17	6.57
DHAN [30]	8.14	6.04	6.37
CANet [31]	8.86	6.07	6.15
AEFNet [32]	7.77	5.56	5.92
CRFormer [33]	**7.32**	5.82	6.07
SEIDNet	7.47	**5.08**	**5.47**

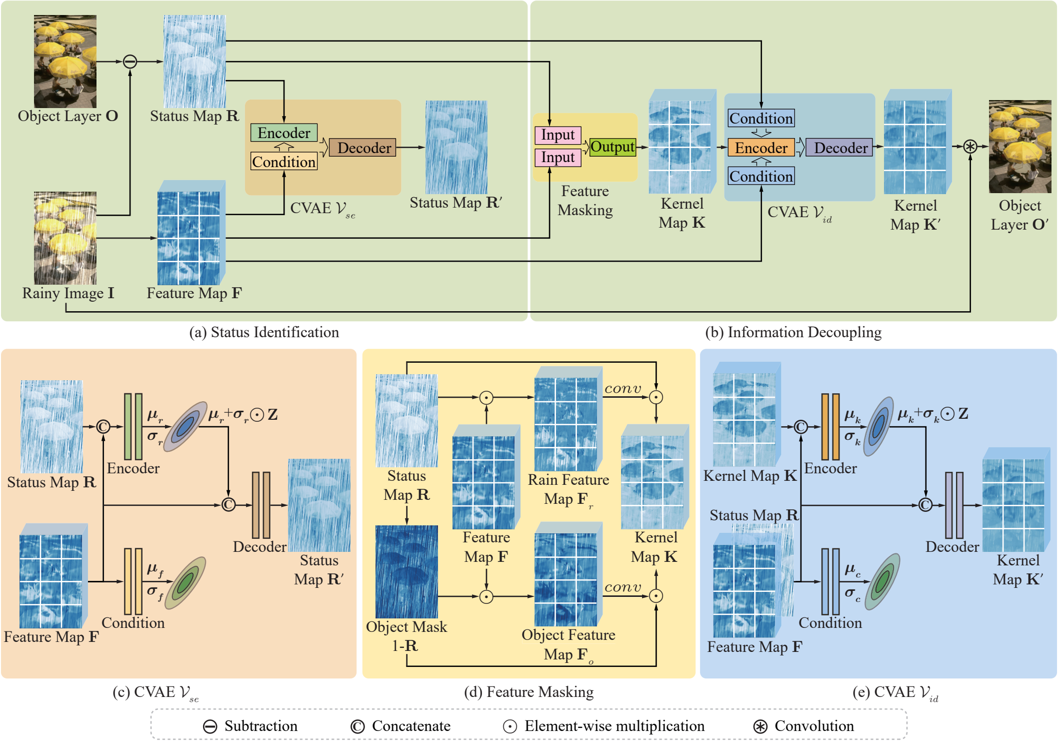
Table 7: We compare SEIDNet with other methods on the test set of ISTD. The performances are reported in term of RMSE.

Method	Shadow	Non-Shadow	All
ST-CGAN [34]	13.4	7.7	8.7
DeshadowNet [35]	15.9	6.0	7.6
Mask-GAN [25]	12.4	4.0	5.3
SP+M-Net [36]	9.7	3.0	4.0
PMDNet [37]	9.7	3.0	4.0
AEFNet [32]	6.5	3.8	4.2
CRFormer [33]	**5.9**	**2.9**	**3.4**
SEIDNet	6.4	3.4	3.9

Table 8: We compare SEIDNet with other methods on the test set of ISTD+. The performances are reported in term of RMSE.

5 Limitation

Quality of Kernels from Feature Masking As evidenced in Table 2, the trivial combination of the discriminative and generative networks increases the network parameters but degrades the performances. It demonstrates that the inaccurate kernels, which are produced by the discriminative network, can harm the performance of the generative network. Though SEIDNet independently generates the kernels for rain removal, it still needs to learn from the kernels that are computed by the discriminative network (i.e., the feature masking in Figure 10) as a reference during training.



(a) Status Identification

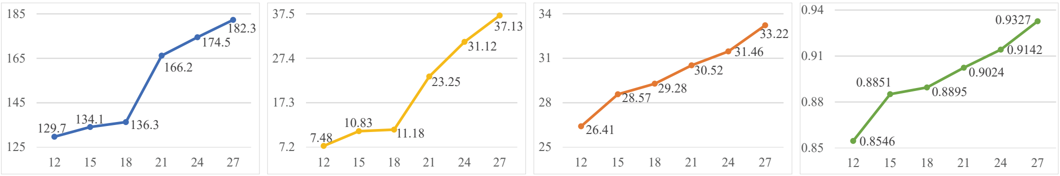
(b) Information Decoupling

(c) CVAE $\mathcal{V}_{se}$

(d) Feature Masking

(e) CVAE $\mathcal{V}_{id}$

Figure 10: 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.



(a) GPU Memory (b) FLOPs (c) PSNR (d) SSIM

Figure 11: Sensitivities of GPU memory in MB (a), FLOPs (b), PSNR (c) and SSIM (d) to the number of layers.

In this section, we evaluate the impact of the kernels, which are produced by the discriminative network, on the deraining performance of SEIDNet. This is done by changing the number of layers of the feature masking. We choose the layer number from the set $\{12, 15, 18, 21, 24, 27\}$, where the feature masking has 27 convolutional layers in the full model. By decreasing the layer number, we reduce the computational overheads (i.e., GPU memory and floating point of operations (FLOPs)), as reported in Figure 11(a–b). On the other hand, fewer layers degrade the deraining performances (see PSNR and SSIM in Figure 11(c-d)). We conjecture that fewer layers weaken the learning capability of the feature masking. In this case, the feature masking likely produces the problematic kernels, which are embedded into the kernel space by CVAE. It misleads the construction of the kernel space, where the problematic kernels are likely sampled for rain removal. We show the visual results of

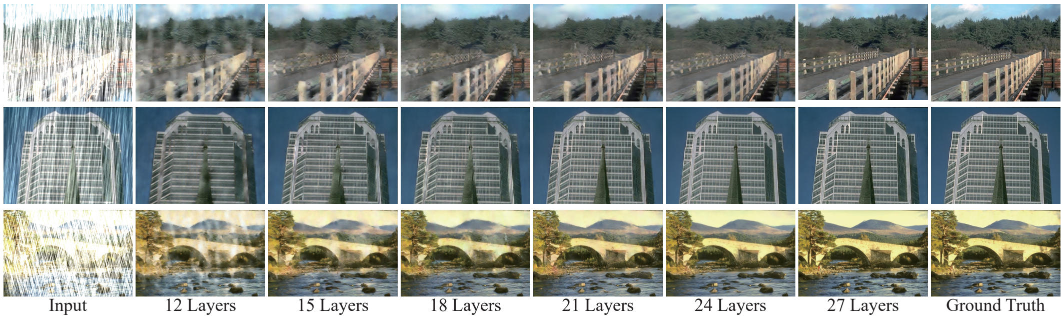


Figure 12: Visual results by changing the number of layers for the feature masking.

using different numbers of layers for the feature masking in Figure 12, where the failure cases are yielded by using fewer layers for computing the reference kernels.

The limitation of our methods motivates the future investigation of how to use CVAE to learn more effective kernels, without depending on the extra and sensitive network for computing the reference kernels.

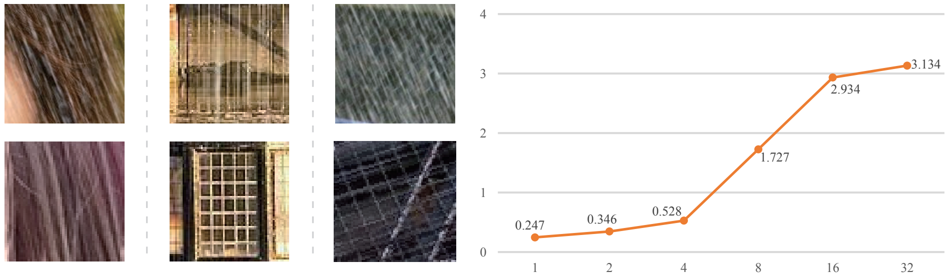
Errors on Some Bright Objects It should be noted that bright objects are extremely similar to the appearances of the rain streaks. The examples of the confusing bright objects and rain streaks can be found in Figure 6: (1) the top-right water region in the first row; (2) the second-left person in white cloth in the fourth row; (3) the bright regions of the wheels in the last row. There are pixels of these bright objects misunderstood as rain streaks, as illustrated in the corresponding status maps. Yet, we use multiple status maps, which provide more differentiable information for separating the bright objects from the rain streaks. Thus, the final deraining results are reasonable. In future work, we plan to further improve the deraining results of the bright objects, while relying on fewer status maps for saving computation.

6 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.

7 Analysis of the Average Kernels

We sample multiple kernels, which are averaged for processing each pixel in the image. Here, we justify the effectiveness of the average kernels in terms of processing the confusing rain streaks and object textures.



(a) image patches

(b) difference of average kernels

Figure 13: The difference of average kernels of the confusing patch pairs, with different. We change the number of sampled kernels $(1, 2, 4, 8, 16, 32)$ for computing the average kernel.

We manually select 100 pairs of image patches from Rain100L and Rain1400 (see Figure 13(a)). Each pair of image patches contain the rain streaks (top row) and the object textures (bottom row), respectively. Each pair of rain and object patches are visually similar. The typical discriminative networks (i.e., EfDeRain, SPDNet, and MPR) compute similar kernels for the confusing rain and object, thus yielding unsatisfactory results in these confusing patches. In contrast, we sample more kernels for each image patch, where the sampled kernels are averaged. We compute the difference (L1 distance) between the average kernels of each pair of confusing rain and object patches. We accumulate and average the differences of all pairs. We change the number of sampled kernels for computing the average kernel. In Figure 13(b), we report the difference of average kernels of the confusing patch pairs. With more sampled kernels, the average kernels are more specific for processing the confusing rain and object patches.

8 Code Segment

We release the implementation of SEIDNet via https://github.com/wxxx1025/SEIDNet.

9 Traing and testing pseudo-code of SEIDNet

Algorithm 1: Training pseudo-code of SEIDNet

```
$epoch = 1$;
while $epoch \leq$ max_epoch do
    Input rainy image $\mathbf{I}$, object layer $\mathbf{O}$ for estimating status map $\mathbf{R}$ via Eq. (3);
    Extract feature map $\mathbf{F}$ from given rainy iamge $\mathbf{I}$;
    {CVAE $\mathcal{V}_{se}$}
    Pass feature map $\mathbf{F}$ and status map $\mathbf{R}$ to CVAE $\mathcal{V}_{se}$;
    Estimate mean value maps and standard deviation maps: $[\boldsymbol{\mu}_r, \boldsymbol{\sigma}_r] \leftarrow encoder([\mathbf{F}, \mathbf{R}])$,
    $[\boldsymbol{\mu}_f, \boldsymbol{\sigma}_f] \leftarrow condition(\mathbf{F})$ in Eq. (4);
    Input $\mathbf{F}$, $\mathbf{Z}$ and $(\boldsymbol{\mu}_r, \boldsymbol{\sigma}_r)$ to the decoder of $\mathcal{V}_{se}$ to generate $\mathbf{R}'$ via Eq. (4);
    Calculate status estimation loss $L_{se}$ via Eq. (5);
    {Feature Masking}
    Estimate kernel maps $\mathbf{K}_r$ and $\mathbf{K}_o$ from $\mathbf{F}$, $\mathbf{R}$ via Eq. (6);
    Estimate kernel map: $\mathbf{K} \leftarrow \mathbf{R} \odot \mathbf{K}_r + (1 - \mathbf{R}) \odot \mathbf{K}_o$ in Eq. (7);
    {CVAE $\mathcal{V}_{id}$}
    Pass $\mathbf{K}$, $\mathbf{F}$ and $\mathbf{R}$ to CVAE $\mathcal{V}_{id}$;
    Calculte mean value maps and standard deviation maps$[\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k] \leftarrow encoder([\mathbf{K}, \mathbf{F}, \mathbf{R}])$,
    $[\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c] \leftarrow condition([\mathbf{F}, \mathbf{R}])$ in Eq. (8);
    Generate kernel map from kernel space: $\mathbf{K}' \leftarrow decoder([\mathbf{F}, \mathbf{R}, \boldsymbol{\mu}_k + \boldsymbol{\sigma}_k \odot \mathbf{Z}])$ in Eq. (8);
    Calculate information decoupling loss $L_{id}$ via Eq. (9);
    {Deraining and overall loss function}
    Employ $\mathbf{K}'$ and $\mathbf{I}$ to estimate object layer $\mathbf{O}'$ via Eq. (10);
    Calcualte deraining loss $L_{de}$ via Eq. (10);
    Calculate overall loss $L$ via Eq. (11);
    Update network weights;
    if $epoch \geq lr\_decrease\_epoch$ then
        Adjust learning rate;
    end if
    $epoch \leftarrow epoch + 1$;
end while
```

Algorithm 2: Testing pseudo-code of SEIDNet

```
$index = 1$
while $index \leq$ len(test_dataset) do
    Extract feature map $\mathbf{F}$ from given rainy iamge $\mathbf{I}$;
    Pass feature map $\mathbf{F}$ to CVAE $\mathcal{V}_{se}$;
    Calculate mean value map and standard deviation map: $[\boldsymbol{\mu}_f, \boldsymbol{\sigma}_f] \leftarrow condition(\mathbf{F})$ in Eq. (12);
    while $m \leq N$ do
        Generate status map from constructed status space: $\mathbf{R}^m \leftarrow decoder([\mathbf{F},\ \boldsymbol{\mu}_f + \boldsymbol{\sigma}_f \odot \mathbf{Z}^m])$ in Eq. (12);
        Pass $\mathbf{F}$, $\mathbf{R}^m$ and $\mathbf{Z}^m$ to CVAE $\mathcal{V}_{id}$;
        Calculate mean value map and standard deviation map: $\boldsymbol{\mu}_c^m, \boldsymbol{\sigma}_c^m \leftarrow condition([\mathbf{F},\ \mathbf{R}^m])$ in Eq. (12);
        Generate kernel map from constructed kernel space:
        $\mathbf{K}^m \leftarrow decoder([\mathbf{F},\ \mathbf{R}^m,\ \boldsymbol{\mu}_c^m + \boldsymbol{\sigma}_c^m \odot \mathbf{Z}^m])$ in Eq. (12);
        $m \leftarrow m + 1$
    end while
    Estimate $\mathbf{K}^u$ via Eq. (13)
    $\mathbf{O} = \mathbf{K}^u \circledast \mathbf{I}$
    $index \leftarrow index + 1$
end while
```

References

[1] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. 9(11), 2008.

[2] Dongwei Ren, Wangmeng Zuo, Qinghua Hu, Pengfei Zhu, and Deyu Meng. Progressive image deraining networks: A better and simpler baseline. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3937–3946, 2019.

[3] 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.

[4] 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 Conference on Computer Vision and Pattern Recognition*, pages 12270–12279, 2019.

[5] 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.

[6] 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.

[7] 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 Conference on Computer Vision and Pattern Recognition*, pages 14821–14831, 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] Qiaosi Yi, Juncheng Li, Qinyan Dai, Faming Fang, Guixu Zhang, and Tieyong Zeng. Structure-preserving deraining with residue channel prior guidance. In *IEEE International Conference on Computer Vision*, pages 4238–4247, 2021.

[10] He Zhang and Vishal M Patel. Density-aware single image de-raining using a multi-stream dense network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 695–704, 2018.

[11] Xueyang Fu, Jiabin Huang, Delu Zeng, Huang Yue, and John Paisley. Removing rain from single images via a deep detail network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3855–3863, 2017.

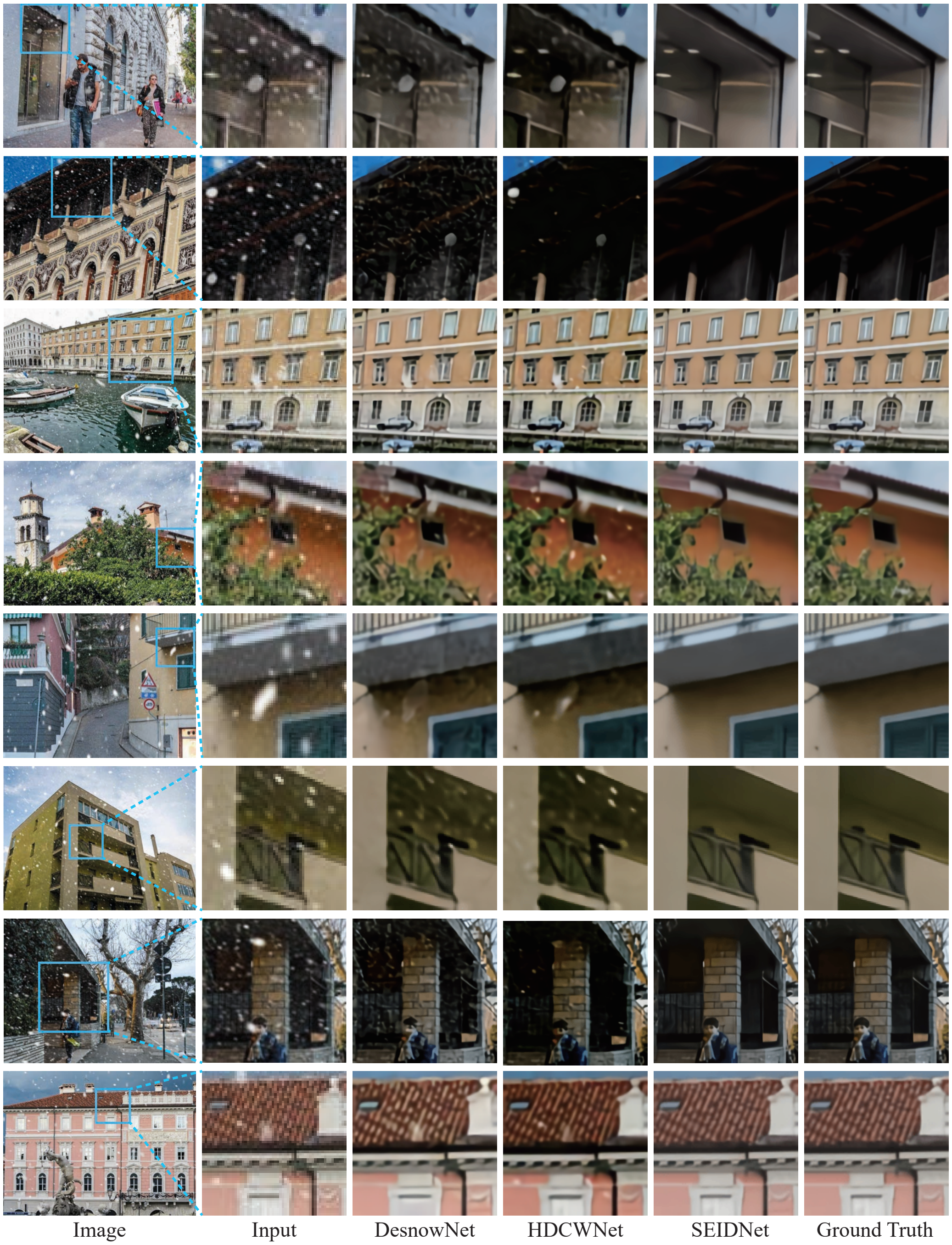


Figure 14: Visual results on the image desnow task. We zoom in the image regions (see the blue rectangles) to compare desnow performance of different methods.

[12] Xing Liu, Masanori Suganuma, Zhun Sun, and Takayuki Okatani. Dual residual networks leveraging the potential of paired operations for image restoration. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7007–7016, 2019.

[13] 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.

[14] 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.

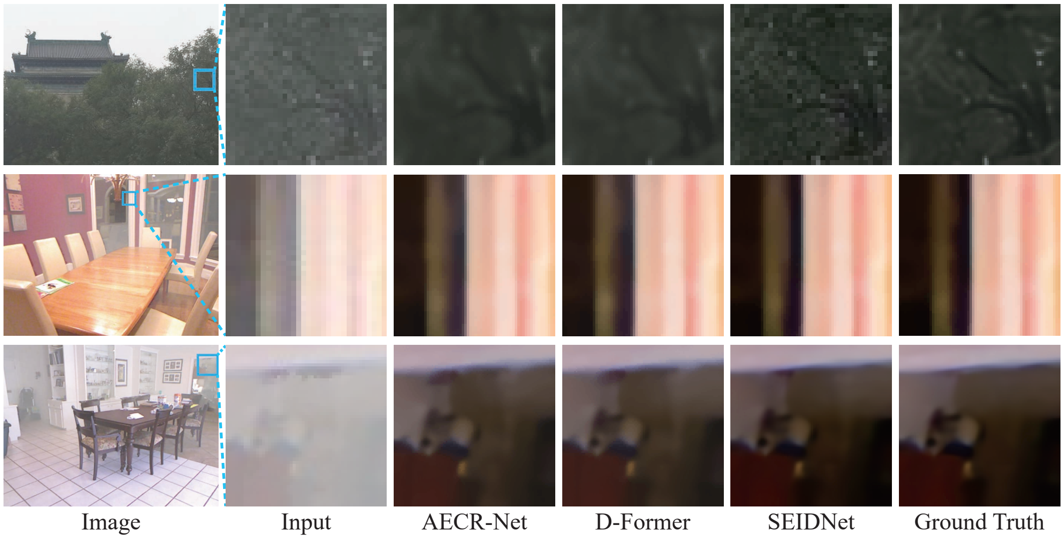


Figure 15: Visual results on the image dehaze task. We zoom in the image regions (see the blue rectangles) to compare dehaze performance of different methods.

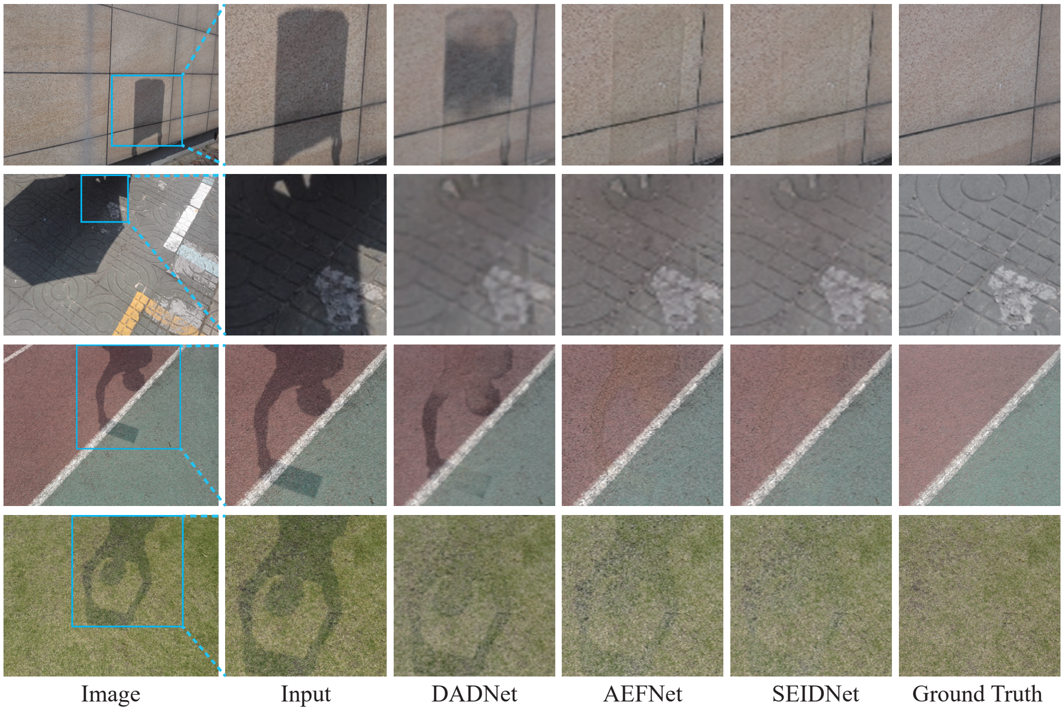


Figure 16: Visual results on the image deshadow task. We zoom in the image regions (see the blue rectangles) to compare deshadow performance of different methods.

[15] 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.

[16] 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 International Conference on Computer Vision*, pages 4196–4205, 2021.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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 Conference on Computer Vision and Pattern Recognition*, pages 2157–2167, 2020.

[22] Xu Qin, Zhilin Wang, Yuanchao Bai, Xiaodong Xie, and Huizhu Jia. Ffa-net: Feature fusion attention network for single image dehazing. In *Association for the Advancement of Artificial Intelligence*, pages 11908–11915, 2020.

[23] 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 Conference on Computer Vision and Pattern Recognition*, pages 10551–10560, 2021.

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

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

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

[27] 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.

[28] Ling Zhang, Chengjiang Long, Xiaolong Zhang, and Chunxia Xiao. Ris-gan: Explore residual and illumination with generative adversarial networks for shadow removal. In *Association for the Advancement of Artificial Intelligence*, pages 12829–12836, 2020.

[29] Zhengxia Zou, Sen Lei, Tianyang Shi, Zhenwei Shi, and Jieping Ye. Deep adversarial decomposition: A unified framework for separating superimposed images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12806–12816, 2020.

[30] Xiaodong Cun, Chi-Man Pun, and Cheng Shi. Towards ghost-free shadow removal via dual hierarchical aggregation network and shadow matting gan. In *Association for the Advancement of Artificial Intelligence*, pages 10680–10687, 2020.

[31] Zipei Chen, Chengjiang Long, Ling Zhang, and Chunxia Xiao. Canet: A context-aware network for shadow removal. In *IEEE International Conference on Computer Vision*, pages 4743–4752, 2021.

[32] 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 Conference on Computer Vision and Pattern Recognition*, pages 10571–10580, 2021.

[33] 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.

[34] Jifeng Wang, Xiang Li, and Jian Yang. Stacked conditional generative adversarial networks for jointly learning shadow detection and shadow removal. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1788–1797, 2018.

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

[36] 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.

[37] Hieu Le and Dimitris Samaras. From shadow segmentation to shadow removal. In *European Conference on Computer Vision*, pages 264–281, 2020.

[38] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[39] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Cvae-gan: fine-grained image generation through asymmetric training. In *IEEE International Conference on Computer Vision*, pages 2745–2754, 2017.

[40] Tianming Wang and Xiaojun Wan. T-cvae: Transformer-based conditioned variational autoencoder for story completion. In *International Joint Conferences on Artificial Intelligence*, pages 5233–5239, 2019.

[41] Danyang Liu and Gongshen Liu. A transformer-based variational autoencoder for sentence generation. In *International Joint Conference on Neural Networks*, pages 1–7, 2019.