Reversing Flow for Image Restoration

Supplementary Material

6. Proof of Proposition 1

Here, we restate Proposition 1 from Sec. 3.1.

Proposition 1. *Given a random process* $\{\boldsymbol{z}_t \,|\, 0 \le t \le 1\}$ *defined by*

$$\frac{\partial \boldsymbol{z}_t}{\partial t} = \boldsymbol{v}(\boldsymbol{z}_t, t); \quad 0 \le t \le 1, \tag{12}$$

where $\boldsymbol{v} \in \mathbb{C}^1$ *is a velocity field, denote the mutual information as* $\mathrm{MI}(\cdot,\cdot)$*, then for any reference random variable* $\boldsymbol{r}$ *and any* $0 \le t_1, t_2 \le 1$*, we have*

$$\mathrm{MI}(\boldsymbol{z}_{t_1}, \boldsymbol{r}) = \mathrm{MI}(\boldsymbol{z}_{t_2}, \boldsymbol{r}). \tag{13}$$

To prove Proposition 1, we need the following lemma which states that the mutual information is invariant to invertible maps.

Lemma 1 ([43])**.** *Given random variables* $\boldsymbol{X}, \boldsymbol{Y}$*, suppose the involved mutual information exists and is finite and* $\boldsymbol{F}, \boldsymbol{G}$ *are two invertible maps, then*

$$\mathrm{MI}(\boldsymbol{X}, \boldsymbol{Y}) = \mathrm{MI}(\boldsymbol{F}(\boldsymbol{X}), \boldsymbol{G}(\boldsymbol{Y})). \tag{14}$$

See the appendix of [43] for a proof of Lemma 1. Now we can prove Proposition 1 using Lemma 1.

Proof of Proposition 1. Suppose Eq. (12) has unique solutions, and the involved mutual information exists and is finite. Consider the flow $\boldsymbol{\Phi}$ of Eq. (12), that is, $\boldsymbol{\Phi}(\boldsymbol{z}, s, t)$ solves the following initial-value problem:

$$\begin{cases} \frac{\partial \boldsymbol{z}_t}{\partial t} = \boldsymbol{v}(\boldsymbol{z}_t, t), \quad 0 \le t \le 1 \\ \boldsymbol{z}_s = \boldsymbol{z} \end{cases}, \tag{15}$$

Symmetrically, consider the following ODE in reverse time:

$$\frac{\partial \boldsymbol{z}'_t}{\partial t} = -\boldsymbol{v}(\boldsymbol{z}'_t, 1-t); \quad 0 \le t \le 1, \tag{16}$$

its flow is denoted as $\boldsymbol{\Psi}(\boldsymbol{z}, s, t)$ which solves the following initial-value problem:

$$\begin{cases} \frac{\partial \boldsymbol{z}'_t}{\partial t} = -\boldsymbol{v}(\boldsymbol{z}'_t, 1-t), \quad 0 \le t \le 1 \\ \boldsymbol{z}'_s = \boldsymbol{z} \end{cases}, \tag{17}$$

Consider $\boldsymbol{u}_t = [\boldsymbol{z}_t^\mathsf{T}, t]^\mathsf{T}$, it satisfies the following autonomous (time-invariant) system:

$$\frac{\partial}{\partial t}\boldsymbol{u}_t = \frac{\partial}{\partial t}\begin{bmatrix}\boldsymbol{z}_t \\ t\end{bmatrix} = \begin{bmatrix}\boldsymbol{v}(\boldsymbol{z}_t, t) \\ 1\end{bmatrix} = \begin{bmatrix}\boldsymbol{v}(\boldsymbol{u}_t) \\ 1\end{bmatrix}. \tag{18}$$

Let $\boldsymbol{U}$ be the flow of Eq. (18), then, $\boldsymbol{U}(\boldsymbol{z}, 0, t) = [\boldsymbol{\Phi}^\mathsf{T}(\boldsymbol{z}, 0, t), t]^\mathsf{T}$. Symmetrically, consider $\boldsymbol{w}_t = [\boldsymbol{z}_t'^\mathsf{T}, 1-t]^\mathsf{T}$ satisfying the following autonomous system:

$$\frac{\partial}{\partial t}\boldsymbol{w}_t = \frac{\partial}{\partial t}\begin{bmatrix}\boldsymbol{z}'_t \\ 1-t\end{bmatrix} = \begin{bmatrix}-\boldsymbol{v}(\boldsymbol{z}'_t, 1-t) \\ -1\end{bmatrix} = -\begin{bmatrix}\boldsymbol{v}(\boldsymbol{w}_t) \\ 1\end{bmatrix}. \tag{19}$$

Let $\boldsymbol{W}$ be the flow of Eq. (19), then, $\boldsymbol{W}(\boldsymbol{z}, 0, t) = [\boldsymbol{\Psi}^\mathsf{T}(\boldsymbol{z}, 0, t), 1-t]^\mathsf{T}$. Comparing Eqs. (18) and (19), it can be seen that they differ only in the sign of the right-hand side. As a result, solving Eq. (18) obtains the *same trajectory in reverse time* as solving Eq. (19) as long as their initial conditions are compatible, that is, $\boldsymbol{z}'_s = \boldsymbol{\Phi}(\boldsymbol{z}, 0, s)$. Consequently, Φ and Ψ form a pair of invertible maps for $0 \le s < t \le 1$:

$$\boldsymbol{\Phi}(\boldsymbol{\Psi}(\boldsymbol{z}, 1-t, 1-s), s, t) = \boldsymbol{\Phi}(\boldsymbol{\Phi}(\boldsymbol{z}, t, s), s, t) = \boldsymbol{z}. \tag{20}$$

For any reference random variable $\boldsymbol{r}$ and any $0 \le t_1, t_2 \le 1$, $\boldsymbol{G} := \boldsymbol{\Phi}(\cdot, t_1, t_2)$ is an invertible map according to Eq. (20). Take $\boldsymbol{G}$ to be identity map, $\boldsymbol{X} = \boldsymbol{r}$ and $\boldsymbol{Y} = \boldsymbol{z}_{t_1}$ in Lemma 1 gives Eq. (14). □

7. Derivation of Entropy-Preserving Degradation Schedule

In Sec. 3.2, we propose the following entropy-preserving degradation schedule:

$$\alpha_t^{\boldsymbol{x}} = 1 - t, \quad \sigma_t^{\boldsymbol{x}} = 1 - \alpha_t^{\boldsymbol{x}} \tag{21}$$

$$\sigma_t^{\boldsymbol{y}} = \beta \cdot (1 - t + \beta)^{-1}, \quad \sigma_t^{\boldsymbol{y}} = 1 - \alpha_t^{\boldsymbol{y}}, \tag{22}$$

where $\beta = 10$ is a hyperparameter. The intuition is that the entropy should remain constant for a reversible process. For a discrete state space, this intuition holds because, according to Proposition 1, for all $0 \le s < t \le 1$ we have:

$$\begin{aligned} H(\boldsymbol{z}_s) &= \mathrm{MI}(\boldsymbol{z}_s, \boldsymbol{z}_s) && \text{(Property of entropy)} && (23)\\ &= \mathrm{MI}(\boldsymbol{z}_t, \boldsymbol{z}_s) && (\boldsymbol{r} = \boldsymbol{z}_s \text{ in Eq. (13)}) && (24)\\ &= \mathrm{MI}(\boldsymbol{z}_s, \boldsymbol{z}_t) && \text{(Symmetry of MI)} && (25)\\ &= \mathrm{MI}(\boldsymbol{z}_t, \boldsymbol{z}_t) && (\boldsymbol{r} = \boldsymbol{z}_t \text{ in Eq. (13)}) && (26)\\ &= H(\boldsymbol{z}_t) && \text{(Property of entropy).} && (27) \end{aligned}$$

As approximations, assume that 1) HQ images follow uniform distribution $\mathcal{U}$ on $[0, 1]$, which is the maximum-entropy distribution on the normalized pixel range; 2) LQ images follow the Dirac distribution at 0, which can be seen as the extreme of image degradations. Then, according to Eqs. (3) to (5) and $\boldsymbol{y}_0 = \mathbf{0}, \boldsymbol{y}_1 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, we have

$$
\begin{aligned}
H(\boldsymbol{z}_t) &= H(\boldsymbol{x}_t) + H(\boldsymbol{y}_t) && (28)\\
&= H(\alpha_t^{\boldsymbol{x}}\boldsymbol{x}_0 + \sigma_t^{\boldsymbol{x}}\boldsymbol{x}_1) + H(\alpha_t^{\boldsymbol{y}}\boldsymbol{y}_0 + \sigma_t^{\boldsymbol{y}}\boldsymbol{y}_1) && (29)\\
&= H((1-t)\boldsymbol{x}_0) + H(\sigma_t^{\boldsymbol{y}}\boldsymbol{y}_1) && (30)\\
&= d\ln(1-t) + \frac{d}{2}(1 + \ln(2\pi)) + d\ln\sigma_t^{\boldsymbol{y}}, && (31)
\end{aligned}
$$

where d is the channel dimension. The last equality comes from the fact that the entropy of $\mathcal{U}[a, b]$ is $\ln|b-a|$ and that the entropy of $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is $\frac{d}{2}(1 + \ln(2\pi)) + \frac{1}{2}\ln|\boldsymbol{\Sigma}|$ [43]. Similarly,

$$
H(\boldsymbol{z}_s) = d\ln(1-s) + \frac{d}{2}(1 + \ln(2\pi)) + d\ln\sigma_s^{\boldsymbol{y}}. \quad (32)
$$

Plugging Eqs. (31) and (32) into Eq. (27) gives

$$
\ln(1-t) + \ln\sigma_t^{\boldsymbol{y}} = \ln(1-s) + \ln\sigma_s^{\boldsymbol{y}}. \quad (33)
$$

Let $s = 0$ and $\sigma_0^{\boldsymbol{y}} = \beta > 0$ in Eq. (33) gives the form of $\sigma_t^{\boldsymbol{y}}$ as:

$$
\sigma_t^{\boldsymbol{y}} = \beta/(1-t). \quad (34)
$$

Eq. (34) is singular at $t = 1$, so we introduce β into the denominator of Eq. (34), obtaining the final form in Eq. (22). Fig. 6 shows the curves of the degradation schedule $\sigma_t^{\boldsymbol{y}}$ under various β values. We fix $\beta = 10$ for all experiments without tuning.

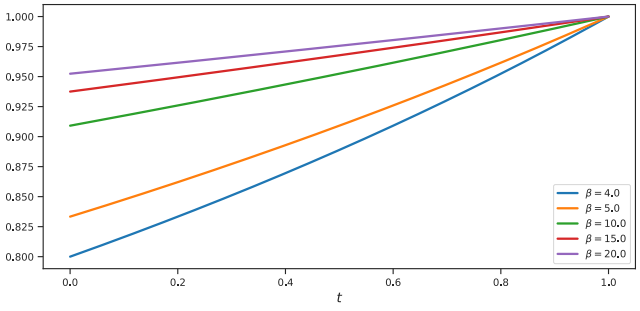

Figure 6. Degradation schedule $\sigma_t^{\boldsymbol{y}}$ with various β values.

Tab. 6 compares our degradation schedule with the constant schedule and the linearly decaying schedule. Our degradation schedule is consistently better than both alternatives, demonstrating the empirical effectiveness of our approach.

Table 6. Experiments about degradation schedules. The average performance on deraining/desnowing/denoising datasets is reported.

Method	PSNR	SSIM
Constant	35.69	0.943
Linear	36.25	0.947
Ours	36.82	0.949

8. Datasets

We evaluate the performance of image restoration methods on five major image restoration tasks, including desnowing, draining, dehazing, denoising, and JPEG compression artifact removal, using synthetic and real-world datasets. Details of the datasets are given below according to their corresponding tasks:

Image Desnowing: Snow100K [60] is a synthetic snow removal public dataset that includes synthetic snow images and corresponding snow-free GT images. The simulated snowflake particles contain a variety of different nozomura, and also have different densities, shapes, trajectories, and transparency in order to add variation. We used Snow100K-l, which has the highest level of diversity, for the evaluation method. Snow100K [60] contains 50000 images for training and 50000 images for testing. RealSnow [126] is a real-world snow removal dataset that acquires image pairs from background-static video. These images feature a variety of urban and natural background scenes that contain varying densities of snowfall and illuminations. RealSnow [126] contains 61500 (crops) and 240 training and testing images.

Image Deraining: Outdoor-rain [49] is a synthetic rain removal dataset, which render synthetic rain streaks and rain accumulation effects based on the provided depth information. These effects include the veiling effect caused by the water particles, as well as image blur. Outdoor-Rain is a set of outdoor rainfall datasets created on clean outdoor images. Outdoor-Rain [49] contains 8100 images for training and 900 images for testing;. LHP [28] is a real-world rain removal dataset. Real image pairs are acquired by keeping the camera motionless to record real rain videos with static backgrounds, which contains a variety of rainfall patterns rain a variety of typical scenarios. LHP [28] contains 300 images for testing.

Image Dehazing: Dense-Haze [4] is a synthetic defogging dataset with dense and homogeneous haze scenes. It contains haze images and corresponding clean images of various outdoor scenes. Dense-Haze [4] contains 49 images for training and 6 images for testing. NH-HAZE [5] is a realistic image dehazing dataset with non-homogeneous hazy and haze-free paired images. The non-homogeneous haze has been generated using a professional haze generator that im-

itates the real conditions of haze scenes. It contains various outdoor scenes. NH-HAZE [5] contains 49 images for training and 6 images for testing, totaling 55 outdoor scenes.

Real Denoising: SIDD [1] is a realistic denoising dataset, which captured real noisy images using five representative smartphone cameras and generated their ground truth images. SIDD [1] contains 288 images for training and 32 images for testing.

Defocus Deblur: DPDD [2] is a synthetic defocus deblurring dataset, which capture a pair of images of the same static scene at two aperture sizes which are the maximum (widest) and minimum (narrowest) apertures possible for the lens configuration. Focus distance and focal length differ across captured pairs in order to capture a diverse range of defocus blur types. DPDD [2] contains 350 images for training and 76 images for testing.

JPEG Artifact removal: The training dataset is collected from DIV2K and FLICKR2K [3] containing 900 and 2650 images, respectively. The testing dataset includes LIVE1 [87] which contains 29 testing images, and BSD500 [6] which contains 500 testing images.

9. Implementation Details

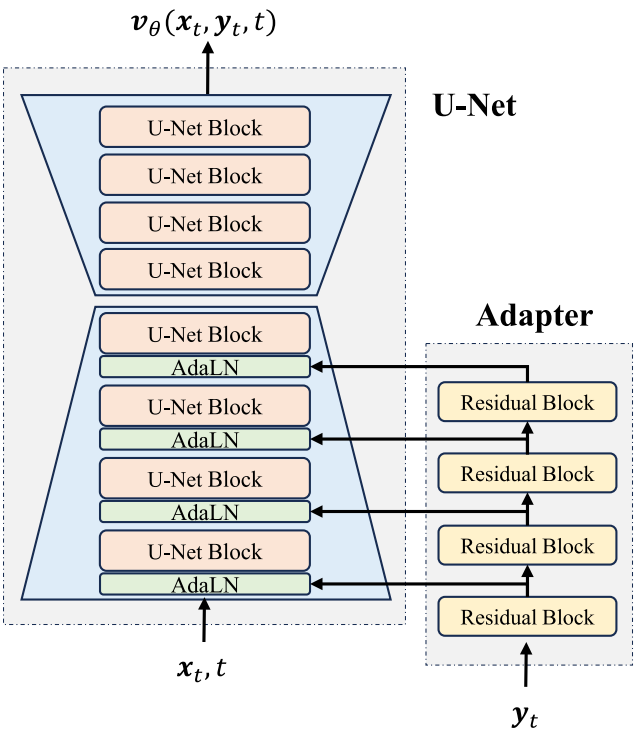

Figure 7. Model architecture.

All experiments adopt the same U-Net architecture from [31] as the backbone. The input to the U-Net is the $\boldsymbol{x}_t$ starting from $\boldsymbol{x}_1$ as the LQ image. The output of the U-Net is the velocity $\boldsymbol{v}_\theta(\boldsymbol{x}_t, \boldsymbol{y}_t, t)$. We remove the class-label conditioning and condition the model on $\boldsymbol{y}_t$ via an adapter [73] as illustrated in Fig. 7. The adapter processes $\boldsymbol{y}_t$ with a stack of residual blocks [29], each downsampling the feature maps of $\boldsymbol{y}_t$ to align with the spatial size of the feature maps of the corresponding in the U-Net. The feature maps are fused with the corresponding feature maps of the U-Net by AdaLN [33]. The output layer of each residual block is initialized to zero, so the overall model is equivalent to the U-Net initially.

The model is trained with a batch size of 8. We use the AdamW optimizer [61] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate is 1e-4 and decayed to 1e-6 via cosine annealing [62]. The input images are normalized to the range of $[-1, 1]$ and randomly cropped to 256×256 during training. The model is trained on each dataset with eight NVIDIA A100 GPUs for 400K iterations.

10. Discussion of Different Approaches

Image restoration is modeled as paired image-to-image translation from LQs to HQs. Historically image restoration methods are *one-step* models that map LQs to HQs with one single step of inference. Diffusion-based models, however, are *multi-step* models that map refine LQs in one or more iterative steps. Similar to diffusion-based methods, our ResFlow allows multistep image restoration that both improves the quality of restored HQ and boosts training (because each step only requires partially restoring the image). Different from diffusion-based methods, ResFlow models HQ-to-LQ degradation by deterministic paths (normalizing flows), which is both easier to learn and more efficient to sample from LQ to HQ. CDPMS [74], RDDM [57], etc. extend diffusion models by modifying the starting point or diffused variables (e.g. introducing residuals). However, they still suffer from low training and sampling efficiency due to stochastic degradation paths of diffusion models. Our ResFlow introduces deterministic degradation paths to solve this problem and achieves superior performance. Note that all the methods we discuss and compare with, including our ResFlow, are learned by supervised training using paired LQ and HQ images. Learning image restoration models with unpaired images is still an open problem.

An interesting method that draws similarity with our ResFlow is Cold Diffusion [10], which tackles image generation. Cold Diffusion also inverts the degradations applied to the images that it is trained to generate (similar to the HQ images). However, the crucial difference between Cold Diffusion and ResFlow is that Cold Diffusion only requires the generated images to be natural, while ResFlow focuses on LQ-to-HQ restoration and also require HQ to preserve LQ's information. The "generation paths" of Cold Diffu-

sion are bounded only on one end; but the degradation paths of ResFlow (Equation (1)) are bounded by both LQ and HQ. Put the difference in implementation, during training, during training, Cold Diffusion learns to estimate the noise-free images; while ResFlow learns to estimate the velocity of degradation flow and introduces an auxiliary variable to ensure reversibility (Equations (9) and (10)). During inference, Cold Diffusion iteratively estimates noise-free images and degrades them with less intensity; while we solve Equations (1) and (3) from $t = 1$ to 0 by Euler integration (that is, accumulating velocity $\times$ step size).

11. Addition Experimental Results

Computational costs. Diffusion models are notoriously slow because they require dozens or even hundreds of inference steps. However, our ResFlow can generate high quality restored in as few as two or even one step. Compared with diffusion-based models such as WeatherDiff [76], we consistently achieve better performance (e.g., **32.82** vs 28.38 PSNR↑ for deraining) with a significantly lower computational cost of **592.44** vs. 2634.8 GFLOPs↓ and **420.8s** vs. 2488.8s latency↓.

Extra visualizations. We provide more visualization results on the synthesized and real-world datasets as shown in Figs. 8 and 9. Synthesized datasets contain Desnowing, Deraining, Dehazing, and Single-image Defocus Deblurring results on Snow100K [60], Outdoor-Rain [49], Dense-Haze [4], and DPDD [2] datasets. Real-world datasets contain Dehazing results on NH-HAZE [5], Denoising results on SIDD [1], Deraining results on LHP [28], and Desnowing results on RealSnow [126]. Extra visualizations on DPDD [2] is shown in Fig. 10, our method significantly outperforms Restormer [115] perceptually.

Figure 11 shows the impact of auxiliary variables on the generated results. After optimizing Equations (9) and (10), ResFlow learns a deterministic coupling from the joint distribution of the auxiliary variable and LQ, to that of the HQ. Conceptually, the auxiliary variable is mapped to the "information difference" between HQ and LQ. When there are multiple possible HQs for an LQ, sampling an auxiliary variable and evaluating Equation (6) will produce a unique velocity, leading to a unique HQ, thus disambiguating the velocity and HQ. Figure 11 is an example of how different auxiliary variables (Aux.) lead to different HQ via different velocities, where blue boxes highlight the differences.

Figure 8. Visual results of synthesized datasets.

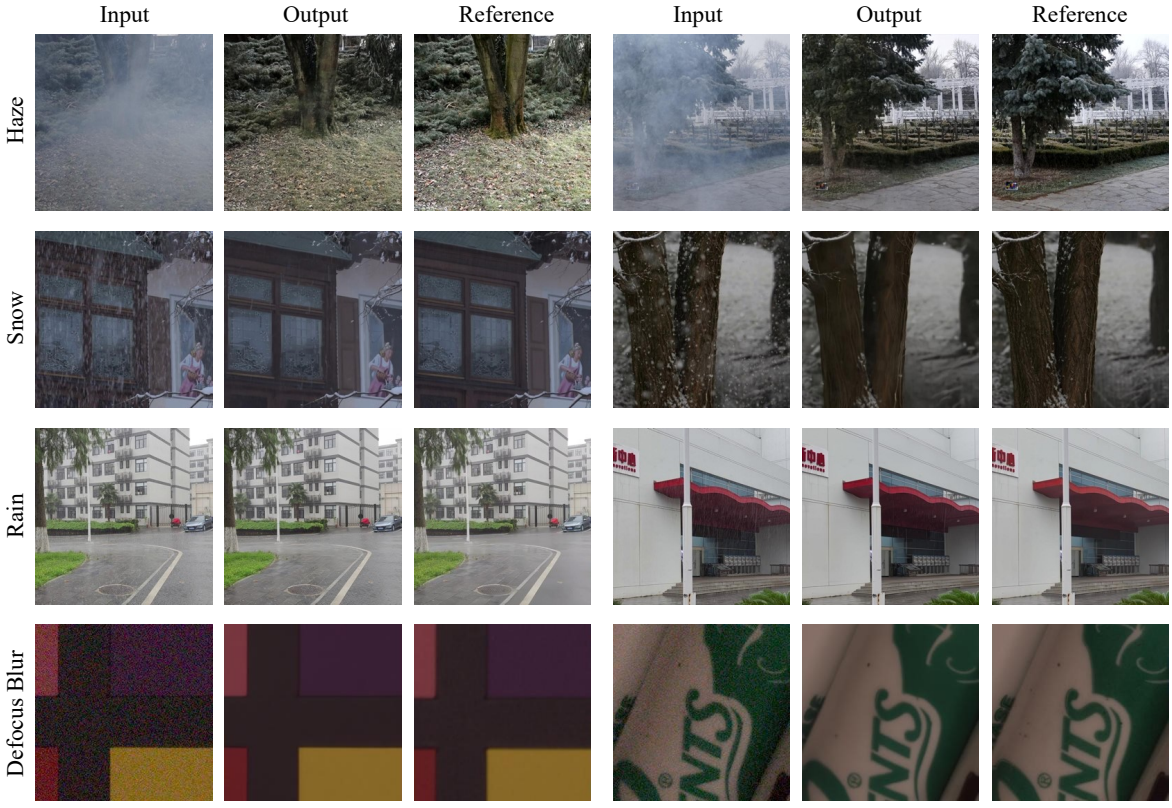

Figure 9. Visual results of real-world datasets.

Figure 10. Extra single-image defocus deblurring results on the DPDD [2] dataset. The part of the image is methodized to observe the local details clearly. From top to bottom: input blurry images, the predicted images obtained by Restormer [115] and our ResFlow.

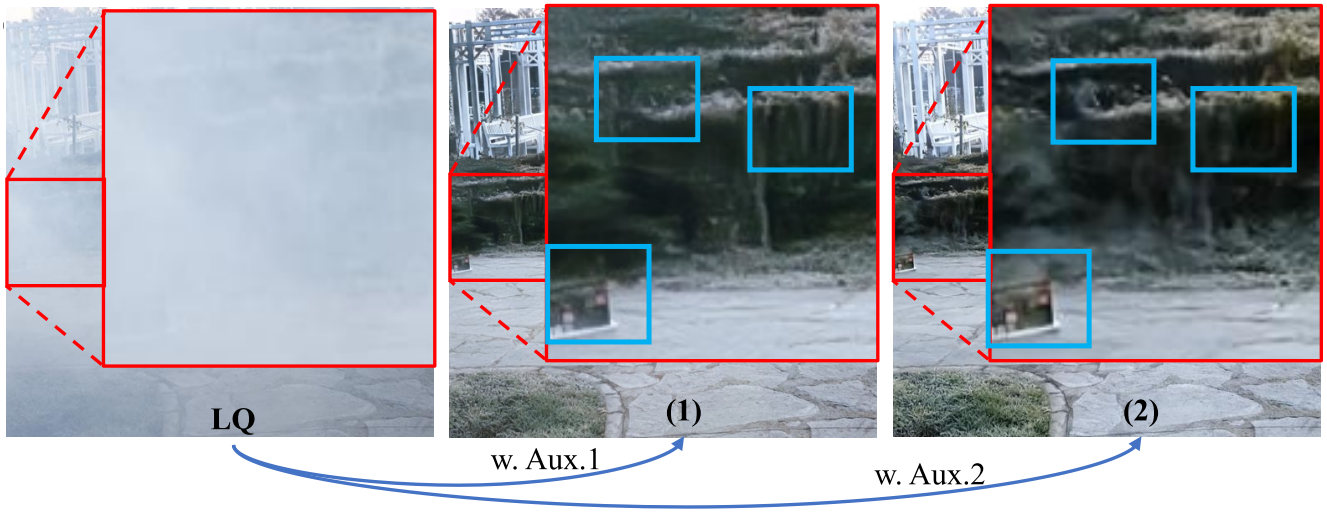

Figure 11. Extra single-image defocus deblurring results on the DPDD [2] dataset. The part of the image is methodized to observe the local details clearly. From top to bottom: input blurry images, the predicted images obtained by Restormer [115] and our ResFlow.