PromptRestorer: A Prompting Image Restoration Method with Degradation Perception —Supplementary Materials—

Cong Wang¹, Jinshan Pan²*, Wei Wang³*, Jiangxin Dong², Mengzhu Wang⁴, Yakun Ju¹, Junyang Chen⁵
¹The Hong Kong Polytechnic University, ²Nanjing University of Science and Technology, ³Dalian University of Technology, ⁴Hebei University of Technology, ⁵Shenzhen University

Abstract

In this supplementary material, we first describe the datasets used in various restoration tasks and experimental details in Sec. 1. Then, we demonstrate the superiority of the proposed PromptDMP compared with commonly used feature fusion module-Spatial Feature Transform (SFT) [27] in Sec. 2. And then, we validate the effectiveness of the used improved ConvNeXt compared with ConvNeXt [21] in Sec. 3. Next, we present the detailed structure of the pre-trained encoder in Sec. 4. Last, we provide additional visual results compared to state-of-the-art approaches on various image restoration tasks in Sec. 5.

1 Datasets Description and Experimental Details

For different restoration tasks, the training and testing datasets are described in Tab. 1.

**Image Deraining:** In accordance with [15, 31, 8], we utilize a composite training dataset consisting of 13,712 image pairs collected from various training datasets, namely Rain100H [30], Rain100L [30], Test100 [34], Test1200 [33], and Test2800 [10]. PromptRestorer is evaluated on Rain100H [30], Rain100L [30], Test100 [34], Test1200 [33], and Test2800 [10]. Our PromptRestorer model is trained for 600 epochs.

**Image Deblurring:** Consistent with recent methods [29, 31], we train PromptRestorer using the GoPro dataset [22]. The GoPro dataset comprises 2,103 blurry/sharp image pairs for training and 1,111 pairs for evaluation. To assess the generalization capacity of our method, we directly apply the GoPro-trained model to the HIDE dataset [24] and the real-world benchmark RSBlur [23]. Our network is trained for 3,000 epochs on the GoPro dataset.

**Image Dehazing:** For dehazing, we train PromptRestorer on the commonly used RESIDE dataset [16]. We employ the ITS dataset for training and use SOTS-indoor as the testing dataset. PromptRestorer is trained for 600 epochs on ITS. Additionally, following [12], we include the real-world datasets Dense-Haze [1] and NH-Haze [2] as experimental datasets, containing 45 training images and 5 testing images, respectively. The network is trained on Dense-Haze and NH-Haze for 3,000 epochs.

**Image Desnowing:** For desnowing, we utilize the CSD [6], SRRS [5], and Snow100K [20] datasets. Following previous works [4], we randomly sample 2,500 image pairs from the training set for training and 2,000 images from the testing set for evaluation. The model is trained for 1,000 epochs on each dataset.

*These authors equally contribute to this work.
Table 1: **Dataset description** for various image restoration tasks.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Datasets</th>
<th>Train Samples</th>
<th>Test Samples</th>
<th>Testset Rename</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deraining</strong></td>
<td>Rain14000 [10]</td>
<td>11200</td>
<td>2800</td>
<td>Test2800</td>
</tr>
<tr>
<td></td>
<td>Rain1800 [30]</td>
<td>1800</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Rain800 [34]</td>
<td>700</td>
<td>100</td>
<td>Test100</td>
</tr>
<tr>
<td></td>
<td>Rain100H [30]</td>
<td>0</td>
<td>100</td>
<td>Rain100H</td>
</tr>
<tr>
<td></td>
<td>Rain100L [30]</td>
<td>0</td>
<td>100</td>
<td>Rain100L</td>
</tr>
<tr>
<td></td>
<td>Rain1200 [33]</td>
<td>0</td>
<td>1200</td>
<td>Test1200</td>
</tr>
<tr>
<td></td>
<td>Rain12 [18]</td>
<td>12</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td><strong>Deblurring</strong></td>
<td>GoPro [22]</td>
<td>2103</td>
<td>1111</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>HIDE [24]</td>
<td>0</td>
<td>2025</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>RSBlur [23]</td>
<td>0</td>
<td>1960</td>
<td>-</td>
</tr>
<tr>
<td><strong>Dehazing</strong></td>
<td>RESIDE/ITS [16]</td>
<td>13990</td>
<td>500</td>
<td>SOTS-Indoor</td>
</tr>
<tr>
<td></td>
<td>Dense-Haze [1]</td>
<td>45</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>NH-Haze [2]</td>
<td>45</td>
<td>5</td>
<td>-</td>
</tr>
</tbody>
</table>

2 **PromptDMP vs. Spatial Feature Transform (SFT) [27]**

Spatial Feature Transform (SFT) [27] which usually generates affine transformation parameters for spatial-wise feature modulation is a commonly used feature fusion module to bridge the feature transformation between conditional features and restoration features [28, 13, 11, 7]. Hence, one may wonder to know whether our proposed PromptDMP is more effective than SFT or not. To validate this, we replace our PromptDMP with SFT to conduct ablation experiments. Tab. 2 shows that our PromptDMP improves the restoration quality by 0.316 dB compared with the results by SFT. It clearly demonstrates that our PromptDMP is more effective than SFT thanks to the designs of global and local prompting perception.

Table 2: **PromptDMP vs. SFT**. Our PromptDMP is more effective than the commonly used SFT [27].

<table>
<thead>
<tr>
<th>Experiment</th>
<th>PSNR</th>
<th>FLOPs (G)</th>
<th>Params (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/ SFT [27]</td>
<td>30.699</td>
<td>108.14</td>
<td>10.36</td>
</tr>
<tr>
<td>w/ PromptDMP (Ours)</td>
<td><strong>31.015</strong></td>
<td>146.36</td>
<td>13.24</td>
</tr>
</tbody>
</table>

3 **Improved ConvNeXt vs. ConvNeXt [21]**

We in this paper employ the improved ConvNeXt by using two depth-wise convolutions with $3 \times 3$ kernel sizes instead of ConvNeXt [21] which uses a depth-wise convolution with $7 \times 7$ kernel sizes. To validate the effectiveness of improved ConvNeXt, we use the ConvNeXt [21] to replace our used improved ConvNeXt, and the results are reported in Tab. 3. It shows that our improved ConvNeXt is more effective. This also reveals that stacking small kernel convolutions can learn better representations than a large kernel convolution with the equivalent receptive field for image restoration.

Table 3: **Improved ConvNeXt vs. ConvNeXt**. Our improved ConvNeXt can help produce high-quality restoration results compared with ConvNeXt [27].

<table>
<thead>
<tr>
<th>Experiment</th>
<th>PSNR</th>
<th>FLOPs (G)</th>
<th>Params (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/ ConvNeXt [21]</td>
<td>30.354</td>
<td>144.28</td>
<td>12.97</td>
</tr>
<tr>
<td>w/ Improved ConvNeXt (Ours)</td>
<td><strong>31.015</strong></td>
<td>146.36</td>
<td>13.24</td>
</tr>
</tbody>
</table>
4 Network Architecture of Pre-trained Encoder

We further specify the architecture of the encoder in Tab. 4. Note that the learnable conditions have
the same encoder architecture as our pre-trained encoder for fair comparisons. The training process
of our pre-trained VQGAN method follows [3].

Table 4: Encoder architecture. The encoder mainly consists of convolution layers, residual
blocks [14], and downsample operation. The encoder gradually encodes input images to spatially-
reduced features.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Encoder1</th>
<th>Downsample</th>
<th>Encoder2</th>
<th>Downsample</th>
<th>Encoder3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape</td>
<td>$48 \times H \times W$</td>
<td>$48 \times \frac{H}{2} \times \frac{W}{2}$</td>
<td>$96 \times \frac{H}{4} \times \frac{W}{4}$</td>
<td>$96 \times \frac{H}{4} \times \frac{W}{4}$</td>
<td>$192 \times \frac{H}{8} \times \frac{W}{8}$</td>
</tr>
<tr>
<td>Blocks</td>
<td>1 Conv &amp; 2 ResBlocks</td>
<td>-</td>
<td>1 Conv &amp; 2 ResBlocks</td>
<td>-</td>
<td>1 Conv &amp; 2 ResBlocks</td>
</tr>
<tr>
<td>Kernel sizes</td>
<td>$3 \times 3$</td>
<td>-</td>
<td>$3 \times 3$</td>
<td>-</td>
<td>$3 \times 3$</td>
</tr>
<tr>
<td>Stride</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>

5 Additional Visual Results

We provide more visual results on various image restoration tasks in Fig. 1-Fig. 38.

Image Deraining Results: Fig. 1-Fig. 4.

Image Deblurring Results: Fig. 5-Fig. 7.

Image Dehazing Results: Fig. 8-Fig. 24.

Image Desnowing Results: Fig. 25-Fig. 38.
Figure 1: Image deraining example on Rain100H [30].
Figure 2: Image deraining example on Rain100H [30].
Figure 3: Image deraining example on Rain100H [30].
Figure 4: **Image deraining** example on Rain100H [30].
Figure 5: **Image blurring** example on GoPro [22].
Figure 6: Image blurring example on GoPro [22].
Figure 7: **Image blurring** example on GoPro [22].
Figure 8: **Image dehazing** example on SOTS-Indoor [16].
Figure 9: Image dehazings example on SOTS-Indoor [16].
Figure 10: Image dehazing example on SOTS-Indoor [16].
Figure 11: **Image dehazing** example on SOTS-Indoor [16].
Figure 12: **Image dehazing** example on SOTS-Indoor [16].
Figure 13: **Image dehazing** example on SOTS-Indoor [16].
Figure 14: **Image dehazing** example on SOTS-Indoor [16].
Figure 15: Image dehazing example on SOTS-Indoor [16].
Figure 16: Image dehazing example on SOTS-Indoor [16].
Figure 17: Image dehazing example on SOTS-Indoor [16].
Figure 18: Image dehazing example on SOTS-Indoor [16].
Figure 19: Image dehazing example on SOTS-Indoor [16].
Figure 20: Image dehazing example on Dense-Haze [1].
Figure 21: **Image dehazing** example on NH-Haze [2].
Figure 22: **Image dehazing** example on NH-Haze [2].
Figure 23: **Image dehazing** example on NH-Haze [2].
Figure 24: **Image dehazing** example on NH-Haze [2].
Figure 25: Image desnowing example on CSD (2000) [6].
Figure 26: Image desnowing example on CSD (2000) [6].
(a) Input

(b) JSTASR [5]

(c) HDCWNet [6]

(d) Uformer [29]

(e) Restormer [31]

(f) PromptRestorer

(g) GT

Figure 27: Image desnowing example on CSD (2000) [6].
Figure 28: Image desnowing example on CSD (2000) [6].
Figure 29: Image desnowing example on CSD (2000) [6].
Figure 30: Image desnowing example on CSD (2000) [6].
Figure 31: Image desnowing example on CSD (2000) [6].
Figure 32: Image de-snowing example on CSD (2000) [6].
Figure 33: Image desnowing example on CSD (2000) [6].
Figure 34: Image desnowing example on CSD (2000) [6].
Figure 35: Image desnowing example on CSD (2000) [6].
Figure 36: Image desnowing example on CSD (2000) [6].
Figure 37: Image desnowing example on CSD (2000) [6].
Figure 38: **Image desnowing** example on CSD (2000) [6].
References


