VRDL HW4: Image Restoration Report

Yi-Hsiang Ho, 111550106

1. Introduction

This task involves image restoration, especially for two types of degradations — Derain and Desnow, using PromptIR [1]. The dataset has 3,200 paired images, with degradation and the corresponding clean images, for training. Both deraining and desnowing task each have 1,600 images, and the degraded types are labeled as well. It also provides 100 images, without degraded type notations, for testing.

The core idea of this work is to change the loss functions to cover more aspects of the clean image, apply test-time augmentation (TTA) to improve performance, and ensemble multiple models to achieve better results. The implementation is based on the official codebase of PromptIR [1]. GitHub repository is available here.

1.1. Image Restoration

Image restoration aims to recover high-quality images from degraded observations, often caused by noise, blur, or environmental factors. Among these tasks, deraining and desnowing are critical for improving visibility and performance in outdoor vision systems. Rain and snow streaks introduce complex, spatially varying distortions that challenge traditional restoration methods.

Figure 1. **The image restoration task.** The first row is deraining task, while the second row is desnowing task.

1.2. PromptIR

PromptIR [1] is a novel framework for blind image restoration that leverages prompt-based learning to adaptively handle various image degradations without requiring explicit degradation labels. By introducing learnable prompts into a frozen image restoration backbone, PromptIR enables flexible and efficient restoration across diverse tasks such as denoising, deblurring, and super-resolution. This approach significantly reduces training costs while maintaining high performance, highlighting the potential of prompt tuning in generalizing image restoration models across different degradation types.

1.3. Test-Time Augmentation

Test-time augmentation (TTA) is a technique used to enhance model performance during inference by applying various transformations, such as flipping, rotation, or scaling, to input images. The model generates predictions for each augmented version, which are then averaged to produce a more robust final output.

1.4. Ensemble Learning

Ensemble learning combines multiple models to improve performance. It averages the predictions of multiple models to make the final prediction. This technique helps to reduce variance, increase accuracy, and improve model robustness.

2. Method

2.1. Data Preprocessing and Augmentation

The training dataset contains 3,200 degraded images with corresponding clean images. It is split into training and validation sets with a ratio of 8:2.

Some data augmentations are applied to the training set. These include:

- Random horizontal flip
- Random vertical flip
- Random rotation
- Random cropping

2.2. Model Architecture

The model follows the same architecture as PromptIR [1], and the pipeline is shown in Fig. 2. It uses a U-Net

style architecture to perform image restoration. An additional Prompt block is added to the model. It consists of a prompt generation module (PGM) and a prompt interaction module (PIM). The prompt generation module generates a prompt based on the latent F_p from input image and prompt components, which is a learnable parameter. The prompt interaction module aims to integrate the prompt into the U-Net pipeline. The "Prompt" is injected only into decoder part of the U-Net.

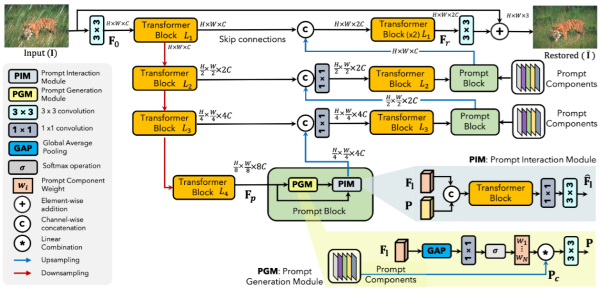

Figure 2. **The pipeline of PromptIR.** (retrieved from PromptIR [1])

2.3. Loss Function

The loss function is a combination of three losses: MSE loss $\mathcal{L}_2$, edge loss $\mathcal{L}_{\text{edge}}$, and perceptual loss $\mathcal{L}_{\text{vgg}}$, while the original implementation only uses L1 loss. The MSE loss is used to measure the pixel-wise difference between the predicted and ground truth images. The edge loss is to make the model learn the edges of the objects in the image. It is calculated by applying Sobel filter to the predicted and ground truth images, and then calculating the L1 loss between the two edge maps. The perceptual loss is calculated by passing the predicted and ground truth images through a pretrained VGG-16 network [2] and calculating the MSE loss between the feature maps of the two images. The final loss is computed as follows:

$$\mathcal{L} = \lambda_1 \cdot \mathcal{L}_2 + \lambda_2 \cdot \mathcal{L}_{\text{edge}} + \lambda_3 \cdot \mathcal{L}_{\text{vgg}} \quad (1)$$

where λ_1, λ_2, and λ_3 are hyperparameters that control the contribution of each loss term. In this task, I set $\lambda_1 = 5.0$, $\lambda_2 = 0.1$, and $\lambda_3 = 0.001$.

2.4. Optimization

For optimization, AdamW [3] is used with an initial learning rate of 2e-4. A linear warmup cosine annealing scheduler [4] dynamically adjusts the learning rate throughout training. The choice of optimizer and scheduler follows the same approach in original PromptIR. The training process is conducted with a batch size of 4 over 300 epochs. The dataset contains 2,560 training images, 640 validation images, and 100 testing images. The best-performing model is selected based on PSNR, which is calculated on validation dataset. The model is trained on a single NVIDIA GeForce RTX 4090 GPU in about 10 hours.

2.5. Inference

Test-time augmentation (TTA) is applied to the model during inference. The model generates predictions for each augmented version of the input image, which includes horizontal flip, vertical flip, and horizontal+vertical flip. The final output is obtained by averaging the predictions of all augmented versions.

Ensemble learning is also applied during inference. Three models trained with different methods are used to generate predictions. The final output is obtained by averaging the predictions of all models. That is, there are in total 4 x 3 = 12 predictions for each image, 4 images for TTA on each model and 3 models selected for ensemble. The three models are:

- Using $\mathcal{L}_2$ and $\mathcal{L}_{\text{edge}}$ as loss function, without $\mathcal{L}_{\text{vgg}}$ and augmentation
- Using all three loss functions, without augmentation
- Using all three loss functions, with augmentation

2.6. Hyperparameters

The hyperparameter details are listed below:

- Learning rate: 2e-4
- Optimizer: AdamW [3]
- Scheduler: Linear warmup cosine annealing [4]
- Batch size: 4
- Epochs: 300
- Patch size: 128
- Loss weights: $\lambda_1 = 5.0$, $\lambda_2 = 0.1$, $\lambda_3 = 0.001$

3. Results

In this section, I compare the performance of each component in the model. The details of each method are listed:

- PromptIR: The original PromptIR model without any modification.
- Aug.: PromptIR + augmentation mentioned in Sec. 2.1
- Loss w/o VGG: PromptIR with the loss function $\mathcal{L}_2$ and $\mathcal{L}_{\text{edge}}$, without $\mathcal{L}_{\text{vgg}}$.
- Loss: PromptIR with all loss function.
- TTA: "Loss" with TTA
- Ensemble: Ensemble of three models mentioned in Sec. 2.5.

The results are shown in Tab. 1. The validation PSNR curve and loss curve are shown in Fig. 3 and Fig. 4, respectively. Each component improves the performance of PromptIR in testing dataset. TTA and ensemble have a significant improvement. The loss for "Loss" is relatively higher since the perceptual loss is not converged yet (see Fig. 5). However, it still performs better than the original PromptIR model.

Method	Val	Test pub.	Test priv.
PromptIR	28.9600	30.1508	29.3773
Aug.	29.2972	30.0079	29.3440
Loss w/o VGG	29.6007	30.3480	29.6707
Loss	28.7450	30.2550	29.6516
TTA	-	30.5405	29.7865
Ensemble	-	**30.7608**	**30.0588**

Table 1. **The PSNR results of different methods.** "Val" refers to validation PSNR. "Test pub." and "Test priv." refer to public and private test set PSNR, respectively. PromptIR is treated as the baseline. If the value is higher than PromptIR, it is highlighted in blue, or it will be in red. The highest values in each column are also highlighted in bold. TTA and Ensemble are not evaluated in validation set, so the values are not available.

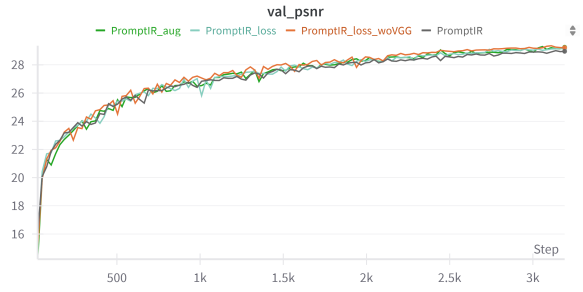

Figure 3. **Validation PSNR curve of different methods.**

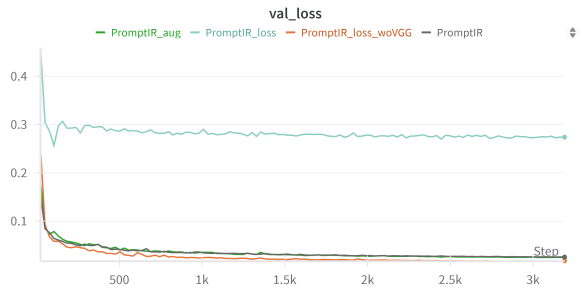

Figure 4. **Validation loss curve of different methods.**

The visualization results of each method are shown in Fig. 6 and Fig. 7. However, since the difference between each method is not significant, it is recommended to see Tab. 1 to evaluate model performance.

Figure 5. **Validation perceptual loss curve for "Loss".**

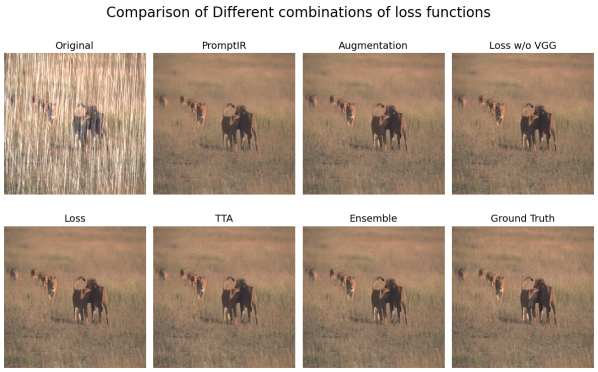

Figure 6. **Visualization results of different methods in deraining task.**

Figure 7. **Visualization results of different methods in desnowing task.**

Other Experiments

The Selection of Scheduler

I've conducted experiments with different learning rate scheduler, including Linear warmup cosine annealing, Co-

sine annealing warm restarts [5], and Cosine annealing. The results are shown in Tab. 2 and visualized in Fig. 8. Even though Warm restarts strategy performs better in validation set (like previous homework), the Linear warmup cosine annealing performs better in both public and private testing. Thus, it was chosen as the scheduler in the model.

Method	Val	Test pub.	Test priv.
Linear warmup	28.9600	**30.1508**	**29.3773**
Warm restarts	**29.1132**	29.8701	29.1540
Cosine annealing	28.8869	29.8357	29.1742

Table 2. **The results of different scheduler.** PSNR is used as the evaluation metric as before. The highest values in each column are highlighted in bold.

Figure 8. **Visualization results of different scheduler.**

Loss Function

Besides the loss functions mention in Sec. 2, I also tried another loss, color loss $\mathcal{L}_{\text{color}}$. This loss is calculated by converting image to LAB color space and calculating the L1 loss between the predicted and ground truth images in the A and B channels. The results are shown in Tab. 3 and Fig. 9.

Method	Val	Test pub.	Test priv.
L1 Loss (PromptIR)	28.9600	30.1508	29.3773
$\mathcal{L}_2 + \mathcal{L}_{\text{edge}}$	**29.3570**	**30.2867**	29.4772
$\mathcal{L}_2 + \mathcal{L}_{\text{edge}} + \mathcal{L}_{\text{vgg}}$	29.1597	30.2547	**29.5248**
$\mathcal{L}_2 + \mathcal{L}_{\text{egde}} + \mathcal{L}_{\text{color}}$	28.9432	29.8923	29.0913

Table 3. **The results of using different combination of loss function.** The highest values in each column are highlighted in bold.

I think it may help to recover the color more accurately, since I find that PromptIR leads to some color distortion in the predicted images. However, adding this color loss even degrades the performance of the model. Looking into the value of color loss (Fig. 10), it seems not converged at all.

Figure 9. **Visualization results of diffenent loss functions.**

It just keeps fluctuating up and down. So I guess this loss may in turn disrupt the optimization process.

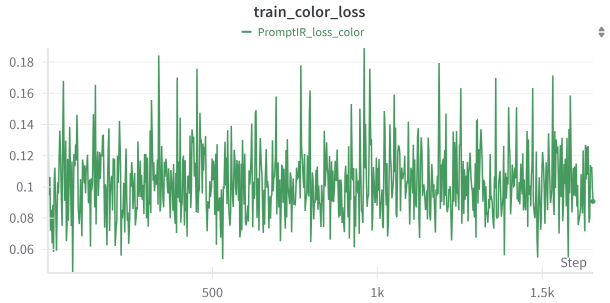

Figure 10. **Color loss curve during training.**

Model Architecture

I've tried to modify the model architecture, including replacing the layer normalization to bias free, and deleting the last transformer block (called "refinement" in their codebase). The results are shown in Tab. 4 and Fig. 11. Even though the model with bias free layer normalization performs better in validation set, it drops its performance in testing set. The final refinement transformer block is not mentioned in the original paper, but it is used in the codebase. Removing this block may lead to the performance drop, so it seems that this block can improve the performance of the model. Thus, the layer normalization and refinement block are kept the same as the original implementation.

Method	Val	Test pub.	Test priv.
PromptIR	28.9600	**30.1508**	**29.3773**
Bias free	**29.3065**	30.0273	29.2934
No refinement	28.8307	29.8706	29.1775

Table 4. **The results of different modification on model architecture.** The highest values in each column are highlighted in bold.

Figure 11. **Visualization results of different model architecture.**

References

[1] Vaishnav Potlapalli, Syed Waqas Zamir, Salman Khan, and Fahad Khan. Promptir: Prompting for all-in-one image restoration. In *NeurIPS*, 2023. 1, 2

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 2

[3] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2017. 2

[4] Linear warmup cosine annealing — lightning-bolts 0.7.0 documentation. 2

[5] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 4