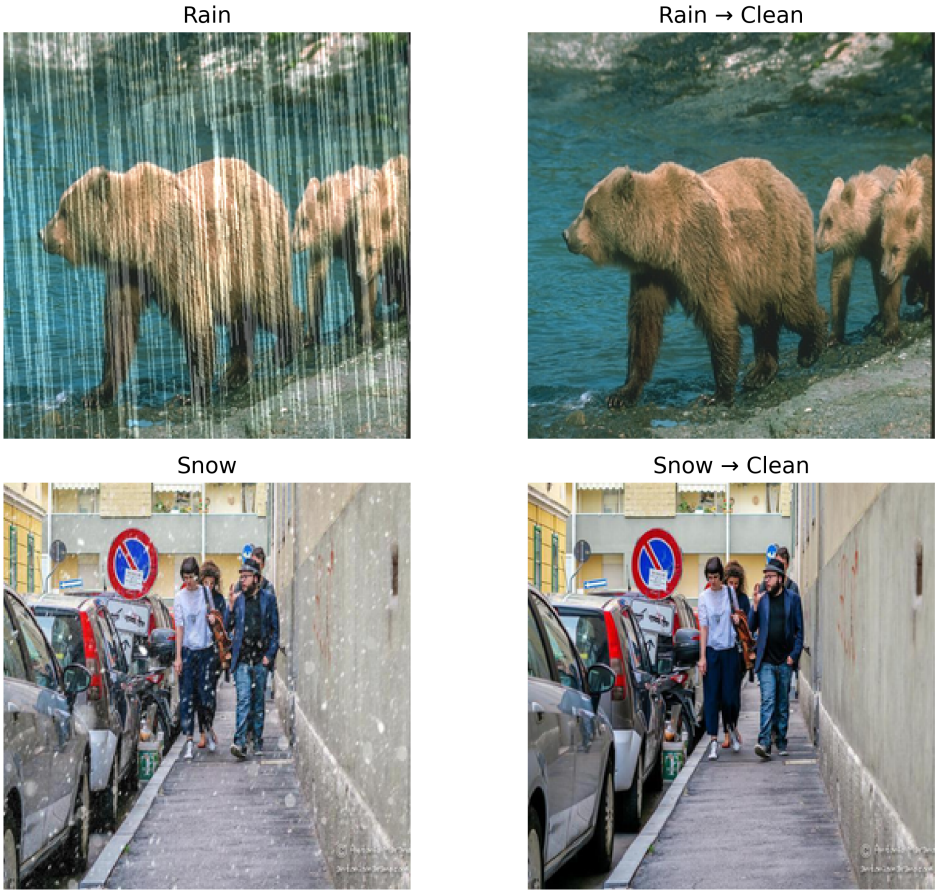
HW4 Image Restoration

王嘉羽
313551052

1 Introduction

This assignment focuses on image restoration, specifically removing rain and snow from degraded images to recover the original clean appearance. The designated model for this task is PromptIR [1], an all-in-one restoration model that automatically determines whether the input image is affected by rain or snow. Due to hardware limitations and the dataset images being 256x256 in size, I conducted most of my experiments using 128x128 images, aiming to achieve comparable results through various training strategies. Since the use of pretrained models was not allowed, one of my key strategies involved generating coarse pretrained weights using the provided dataset, followed by diverse data augmentation techniques to simulate different conditions and perform fine-tuning. Another notable strategy was patch testing: because the final output needs to be 256x256 while my model accepts 128x128 inputs, I divided each image into thirty-six 128x128 patches and used a Gaussian filter for weighted summation to reduce boundary artifacts. As a result, I achieved a PSNR [2] of 30.x on the public dataset, which I believe demonstrates the effectiveness of my approach.

Github: https://github.com/vayne1125/NYCU-Visual-Recognitionusing-Deep-Learning/tree/main/HW4_Image-Restoration



2 Method

I divided the workflow of this assignment into three stages: the first two stages are for training, and the third stage is for testing. In the first stage, I generate pretrained weights to enable the model to initially learn the basic image restoration task. The second stage involves fine-tuning, where the goal is for the model to focus on learning finer details. In the third stage, I perform testing using patch-based testing and test-time augmentation (TTA), applying a Gaussian filter to reduce boundary artifacts during patch merging.

2.1 Stage 1 - Pretrained Weight Generation

Pre-processing

The original dataset images are 256x256 in size. I split each image into four 128x128 patches and fed them into the model together for training.

Model Architecture

I directly used the PromptIR [1] model, which provides all-in-one image restoration capabilities and automatically determines whether the degradation is due to rain or snow. The input and output image sizes are both 128x128.

Validation PSNR Calculation

For validation, each 256x256 image is divided into four 128x128 patches. After inference, the four patches are reassembled into a 256x256 image, which is then compared with the clean ground truth to compute the PSNR [2].

Hyperparameter Settings

Hyperparameter	Value
Optimizer	AdamW
Learning Rate	0.0002
Weight Decay	0.01
Scheduler	linearWarmupCosineAnnealingLR [3]
Warmup Epochs	15
Epochs	50
Batch Size	1
Early Stop	epoch//5
Loss	MSE

Table 1: Hyperparameter Settings

2.2 Stage 2 - Finetuning

Pre-processing

Each 256x256 input image is randomly cropped into a 128x128 patch. The following data augmentation techniques are then randomly applied (possibly multiple at once):

- ❖ Horizontal flip (probability = 0.5)
- ❖ Vertical flip (probability = 0.5)
- ❖ Rotation by 180° (probability = 0.5)

These augmentations help create a dataset different from that used in Stage 1, allowing the model to learn diverse features.

Model Architecture and Validation PSNR Calculation

The model architecture and PSNR [2] validation process are the same as in Stage 1. This stage focuses on adapting the model to the augmented dataset for improved fine-grained learning.

Hyperparameter Settings

Hyperparameter	Value
Optimizer	AdamW
Learning Rate	0.00001
Weight Decay	0.01
Scheduler	linearWarmupCosineAnnealingLR [3]
Warmup Epochs	8
Epochs	80
Batch Size	4
Early Stop	epoch//5
Loss	0.6 * L1 + 0.2 * MSE + 0.2 * SSIM

Table 2: Hyperparameter Settings

2.3 Stage 3 - Testing

In the testing stage, for each input 256x256 image, we adopt a strategy that combines patch-based inference with test-time augmentation (TTA) to generate the final restored result. The detailed procedure is as follows:

1. **Test-Time Augmentation (TTA):** A predefined set of augmentations—including horizontal flip, vertical flip, 180° rotation, and their combinations—is applied to the original 256x256 image. Each augmentation produces a variant of the original image.
2. **Patch Extraction and Model Inference:** For each TTA-transformed image, overlapping patches of size 128x128 are extracted with a stride of 32 pixels, yielding 36 patches per image. These patches are individually fed into the trained model for de-raining.
3. **Image Reconstruction and Fusion:** The model outputs for each patch are reassembled based on their spatial positions to form a complete restored image. In overlapping regions, we apply **Gaussian weighted averaging** to merge pixel values, effectively smoothing the seams and reducing boundary artifacts.
4. **Inverse Transformation:** For each restored image corresponding to a TTA variant, we apply the exact **inverse transformation** to revert it to the original orientation and state.
5. **Final Output Averaging:** All restored images obtained from the inverse-transformed TTA variants are **averaged pixel-wise**. The resulting image is the final restoration output for the given input.

This multi-step process leverages the model's localized inference ability, enhances robustness through TTA, and ensures coherence and quality in the final output via overlapping patch fusion.

3 Additional Experiments

3.1 Effects of Different Training Strategies

To address hardware limitations (processing 256x256 images directly was too resource-intensive), I adopted two strategies to accelerate computation:

1. **Strategy 1 - 128x128 Input with Upsampling Output:** Resize the input image to 128x128 and add an upsampling layer at the end of the PromptIR model to output 256x256. This allows the model to learn the mapping from 128x128 to 256x256 during training.
2. **Strategy 2 - Patch-Based Two-Stage Training:** In Stage 1, split each 256x256 image into four 128x128 patches for coarse training. In Stage 2, randomly crop a 128x128 region from the original 256x256 image and apply data augmentation (e.g., flipping and rotation) for fine-tuning.

These two experiments were designed to handle the memory and computational constraints caused by large input images.

In Strategy 1, I aimed to solve the input-size problem by incorporating an extra upsampling layer so the model could directly learn to upscale features from 128x128 to 256x256.

In Strategy 2, I considered that learning an upsampling layer might be difficult, so I took a different approach: using a two-stage training pipeline. The first stage applies MSE loss to allow the model to roughly capture dataset characteristics. In the second stage, I used data augmentation and L1+SSIM loss for fine-tuning, helping the model focus on more detailed features.

I believe both strategies offer viable ways to reduce training time while still enabling effective model learning under hardware constraints.

3.2 Effects of Different Data Augmentation Techniques

For data augmentation, I mainly applied horizontal and vertical flips. Additionally, I compared the effectiveness of the following three augmentation strategies:

1. **Rotation with Four Directions:** Rotate the images by 0°, 90°, 180°, and 270°.
2. **Rotation with Two Directions:** Rotate the images by 0° and 180°.
3. **Color Jittering:** Randomly change the brightness, contrast, saturation, and hue of the input images.

These augmentations were motivated by the fact that the dataset is relatively small. Therefore, I used data augmentation to increase the diversity of the training samples. I believe that applying data augmentation can effectively improve the model's generalization ability and help it learn more accurate features.

In particular, since rain has directional properties, I focused on comparing different rotation strategies to determine the most suitable augmentation method for this task.

4 Results

4.1 Comparison of Training Strategies

A visual comparison between Strategy 1 and Strategy 2 is shown in Figure 1. Since the training procedures and loss computation methods differ between the two strategies, we do not compare their training loss curves. The validation PSNR and public PSNR results are reported in Table 3.

From the visual comparison, we can observe that in Strategy 1, the regions highlighted by the red boxes are more blurred and the overall restoration quality is worse. I believe this is because learning the transformation from 128x128 to 256x256 via an upsampling layer is challenging under the current hardware constraints.

On the other hand, Strategy 2 shows significant improvements in image quality. I believe this is due to using 128x128 patches, which avoids the errors introduced by resizing or upsampling. In addition, by adopting a two-stage training process—first learning coarse features with MSE loss, followed by fine-tuning with data augmentation and L1+SSIM loss—the model is able to capture finer details.

This staged approach also enables me to experiment with different augmentation techniques in the second stage without retraining the entire model from scratch, making it easier to find a fine-tuning method that suits this task.

Strategy Name	Validation PSNR	Public PSNR
1: 128_to_256	23.82	-
2: 2-Stage Training	28.60	**30.10**

Table 3: Quantitative Comparison of Different Training Strategies

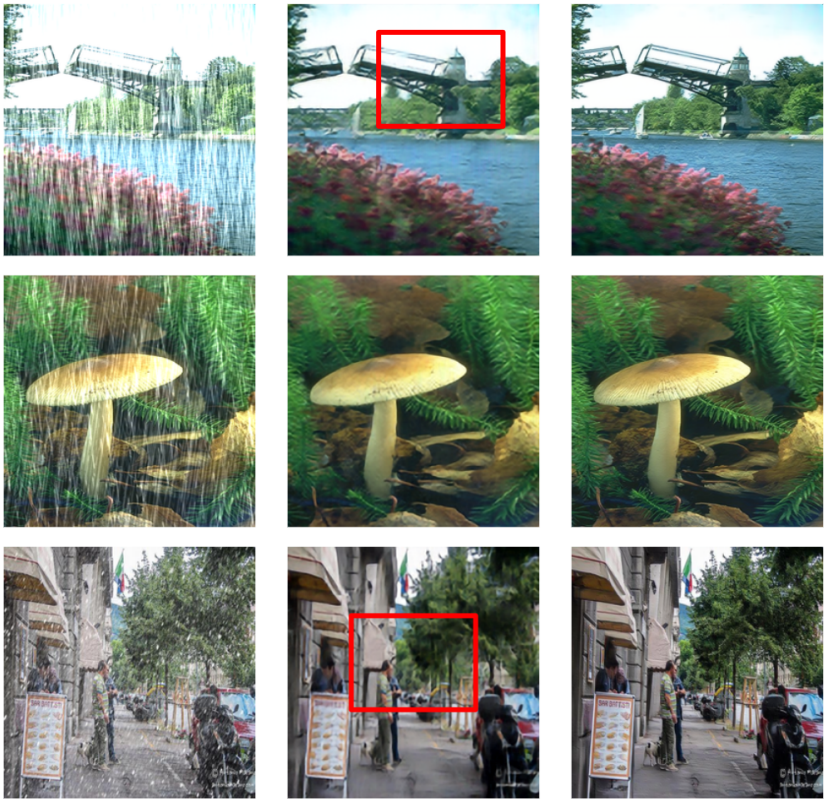


Figure 1: Comparison Results of Training Strategies

4.2 Comparison of Data Augmentation Techniques

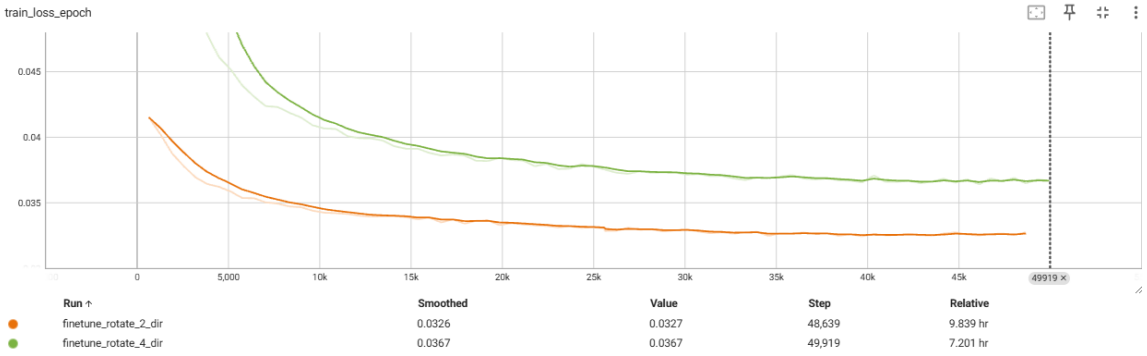
The results for color jittering were notably poor, with validation PSNR dropping by 1 point per epoch. Due to this rapid degradation, I terminated training after only 3 epochs. I suspect this poor performance is due to the nature of the deraining task, where the model relies on subtle color and intensity differences to detect and remove rain streaks. The color jitter operation may have disrupted these cues, making it harder for the model to learn meaningful patterns.

For the comparison between 4-directional rotations (0°, 90°, 180°, 270°) and 2-directional rotations (0°, 180°), the training loss and validation PSNR curves are shown in Figure 2, and the final PSNR values are presented in Table 4.

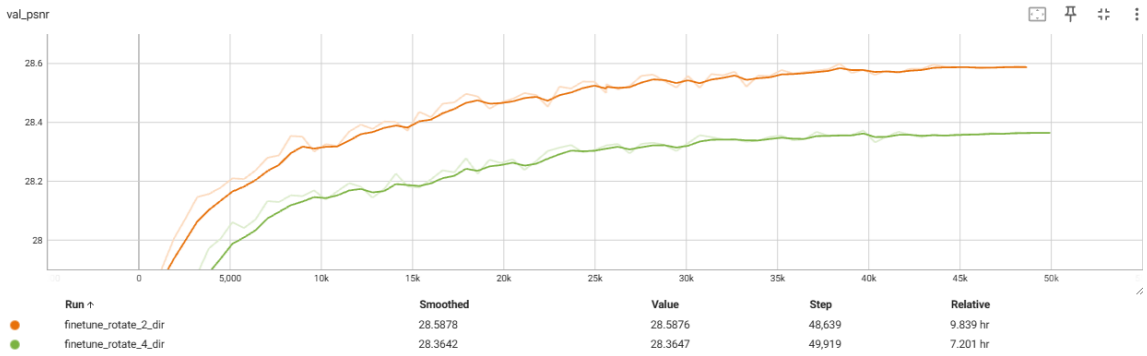
It is evident that the model trained with 2-directional rotation augmentation achieved significantly better results than the one trained with 4-directional rotations. I believe this is because image orientation plays a crucial role in deraining tasks, and applying rotations that distort this directionality may hinder the model' s ability to learn relevant features effectively.

Data Augmentation	Validation PSNR	Public PSNR
4 Dir.	28.36	30.60
2 Dir.	28.60	**31.10**

Table 4: Quantitative Comparison of Different Data Augmentation Techniques



(a) Training loss across experiments.



(b) Validation PSNR across experiments.

Figure 2: Overall comparison of training loss and validation PSNR across different data augmentation strategies.

4.3 Final Results

The final model was based on Strategy 2, combined with data augmentation using 0° and 180° rotations. This configuration achieved a public PSNR of 31.10, which I consider a strong result given the task complexity and limited dataset size.

Figure 3 shows some visualization examples. The top row displays the original rainy images, while the bottom row shows the corresponding derained outputs produced by the model.

Figure 3: Visualization of model outputs. Top: input images; Bottom: derained results.

These results demonstrate the effectiveness of directional-aware data augmentation and the benefits of staged training for deraining tasks.

References

[1] V. Potlapalli, S. W. Zamir, S. Khan, and F. S. Khan, “Promptir: Prompting for all-in-one blind image restoration,” 2023. [Online]. Available: https://arxiv.org/abs/2306.13090

[2] F. A. Fardo, V. H. Conforto, F. C. de Oliveira, and P. S. Rodrigues, “A formal evaluation of psnr as quality measurement parameter for image segmentation algorithms,” 2016. [Online]. Available: https://arxiv.org/abs/1605.07116

[3] I. Loshchilov and F. Hutter, “Sgdr: Stochastic gradient descent with warm restarts,” 2017. [Online]. Available: https://arxiv.org/abs/1608.03983