Visual Recognition Spring 2025 Homework #4

Student ID: 313551099

Student Name: 李以恩

GitHub Repository Link

1. Introduction

Image restoration aims to recover clean images from degraded inputs that suffer from various corruptions such as rain, snow, noise, and blur. In real-world applications, these degradations are often mixed or unknown, posing a significant challenge to traditional single-task restoration methods. The task in this project is to train a single deep learning model capable of restoring images degraded by two types of adverse noise: rain and snow, without prior knowledge of the degradation type at inference.

My approach leverages the PromptIR framework, a novel all-in-one blind image restoration model that introduces trainable prompt vectors to encode different degradation types implicitly. Unlike explicit degradation classification, PromptIR uses these learned prompts to condition the restoration network, allowing it to handle multiple degradations in a unified architecture. This method avoids the overhead and potential error of explicit degradation type prediction while enabling effective restoration across diverse conditions.

---

2. Method

This section describes my data preprocessing pipeline, the model architecture, key hyperparameters, and the training strategies adopted to optimize performance.

2.1 Data Preprocessing and Augmentation

The dataset comprises paired degraded and clean images for rain and snow degradations. Training and validation sets include 3,201 images per degradation type, and the test set contains 50 images per type without labels.

For data preprocessing, we apply the following:

- Images are loaded as RGB and normalized from the range [0, 255] to [-1, 1].
- For training, spatial augmentations include random horizontal and vertical flips and 90-degree rotations, ensuring model robustness to orientation variations.
- Color jittering was experimented with, but was selectively applied based on performance impact.
- To maintain the alignment between degraded and clean images during augmentation, I utilize Albumentations' `additional_targets` feature to apply identical transformations to both inputs.

No external data or pretrained weights are used, adhering strictly to the project constraints.

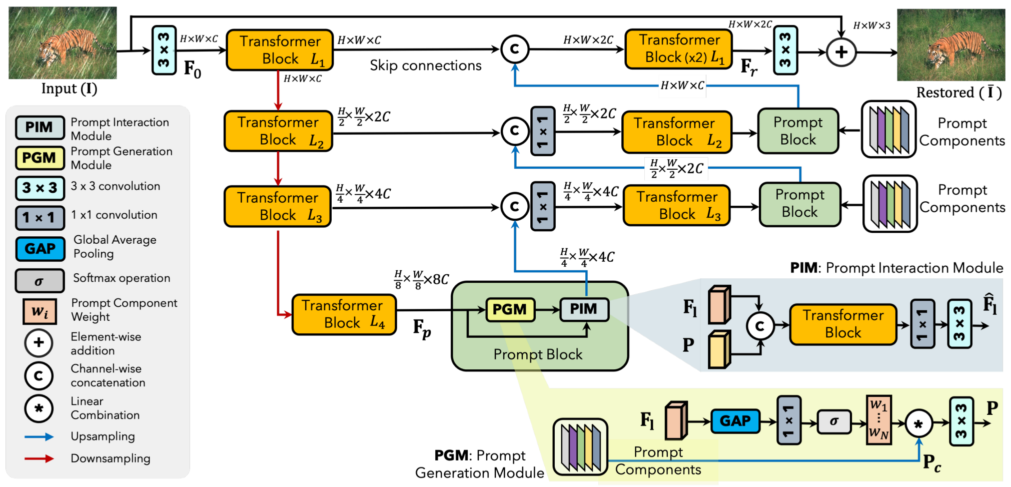
2.2 Model Architecture

I adopt the official PromptIR architecture, which is a transformer-based encoder-decoder network conditioned on learned degradation prompts.

Key components include:

- **OverlapPatchEmbed:** Converts input images into overlapping patches with convolutional projection, preserving spatial context.
- **Transformer Blocks:** Multi-head self-attention layers with gated depthwise convolution feed-forward networks, arranged hierarchically across four encoder and decoder stages.
- **PromptGenBlocks:** Trainable prompt vectors generated per stage, conditioned on latent features, and fused into the decoder to inform restoration conditioned on degradation type.
- **Downsample/Upsample Modules:** Pixel-unshuffle and pixel-shuffle operations for resolution scaling between stages.
- **Noise Level Blocks:** Transformer blocks applied on concatenated feature maps augmented with prompts, designed to refine restoration by explicitly modeling degradation noise.
- **Refinement Blocks:** Additional transformer layers refining the output before the final convolutional reconstruction layer.

The model outputs an RGB image of the same size as input, with residual learning applied by adding the model output to the input degraded image.



2.3 Hyperparameters

My experiments use the following key hyperparameters:

Parameter	Value
Patch size	256 × 256 pixels
Batch size	2 images per GPU
Optimizer	AdamW (initial LR = 2×10^{-4}, weight decay = 1×10^{-4})
Learning rate schedule	5-epoch linear warm-up → cosine annealing decay
Loss function	L1 loss (weight 1.0) + SSIM loss (weight 0.1)
Training epochs	Up to 120; early stopping with patience = 15 epochs
Mixed precision	Enabled with gradient scaling
EMA	Exponential Moving Average starting from epoch 10

2.4 Training Strategy

I train using a two-stage approach:

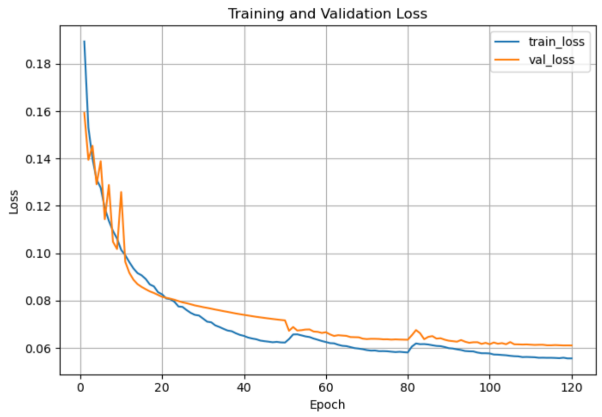
- **Training phase:** The model is trained on augmented paired degraded-clean images, minimizing the combined L1 and SSIM loss, allowing the network to learn perceptually faithful restoration.
- **Validation phase:** At each epoch, the model is evaluated on a held-out validation set using PSNR as the primary metric, enabling early stopping and best model checkpointing based on PSNR improvements.

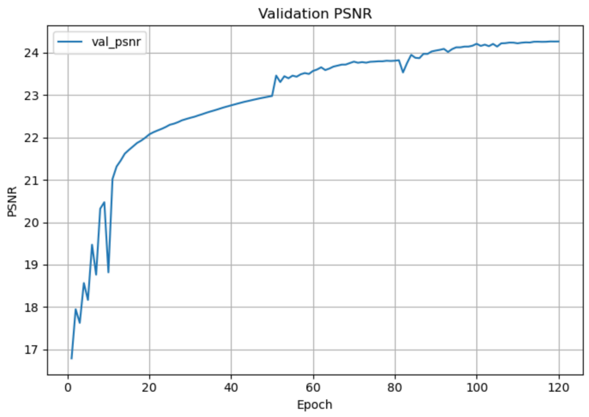
I incorporate EMA for better generalization and smooth convergence. The scheduler uses warm-up to avoid early training instability.

3. Results

- **Training curves**

  My model achieves a validation PSNR of approximately 23.5 dB after 50 epochs, outperforming the baseline PromptIR implementation without warm-up and EMA. The training and validation loss curves indicate steady convergence, with validation PSNR steadily improving over epochs.





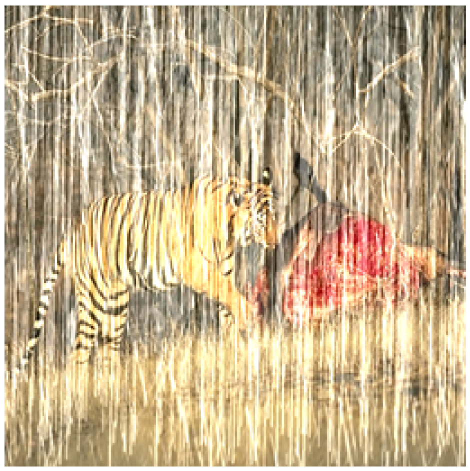
Training and Validation Loss curves showing steady convergence.

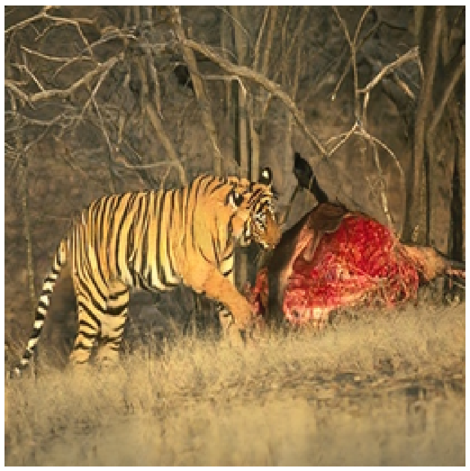
Validation PSNR over epochs, reaching over 24 dB.

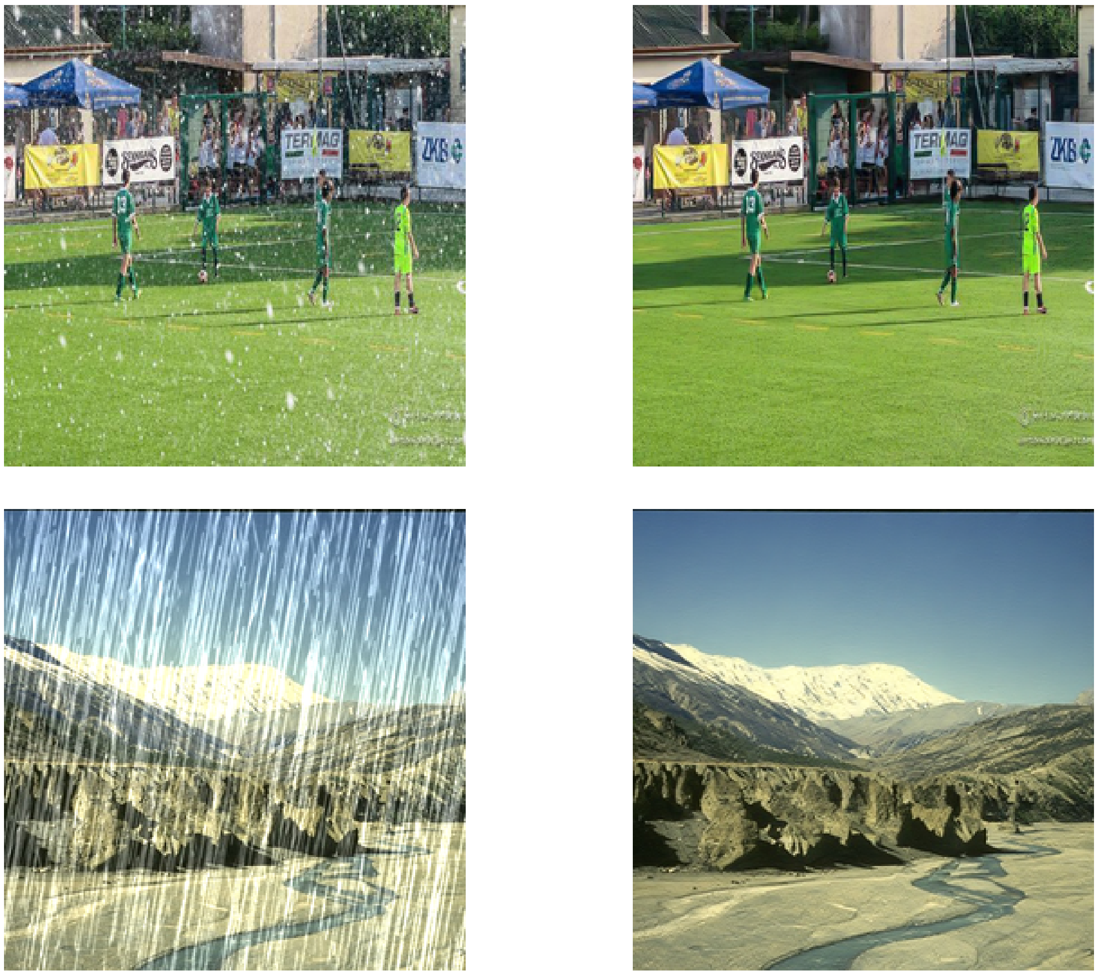
- **Visualizations**

  Qualitative results show effective removal of rain streaks and snow artifacts, producing clean and visually pleasing outputs that maintain texture and edge details.

  - Original image
  - After restoration







4. Experiments

Experiment 1 – Effect of Warm-up and EMA

1. **Hypothesis:** Introducing a 5-epoch linear warm-up in the learning rate schedule combined with EMA stabilization improves convergence and final PSNR.
2. **How it may work:** Warm-up prevents early gradient explosion or suboptimal minima, while EMA smooths weight updates, reducing variance in validation metrics.
3. **Results:** The model with warm-up and EMA achieved a PSNR increase from ~29.95 dB to 30.28 dB at 50 epochs compared to the baseline.

	Baseline (epoch = 50)	Warmup + EMA (epoch = 50)
PSNR	29.95	30.28

4. **Implications:** Proper scheduling and weight averaging techniques can significantly enhance model generalization and training stability.

Experiment 2 – Data Augmentation Variants

1. **Hypothesis:** Including spatial augmentations (rotation, flip) and color jittering affects the model's robustness and final performance.
2. **How it may work:** Rotation and flips improve invariance to image orientation, while color jitter introduces color variability reflecting real-world conditions.
3. **Results:** Models trained without rotation or flip but with color jitter reached 28.92 dB; with neither augmentation 29.11 dB; and with both 29.05 dB PSNR.

	e1: patch_size = 256 - no rotation or flip - with color jitter	e2: patch_size = 256 - no rotation or flip - no color jitter	e3: patch_size = 256 - with rotation or flip - with color jitter
PSNR	28.92	29.11	29.05

4. **Implications:** Spatial augmentations had a minor effect on performance, but color jitter slightly affected results, indicating that augmentation strategy should be carefully tuned.

Experiment 3 – Patch Size Effects

1. **Hypothesis:** Training with smaller patches (128) versus larger patches (256) impacts convergence and model capability to capture context.
2. **How it may work:** Smaller patches reduce GPU memory use but limit spatial context; larger patches provide more global information at a computational cost.
3. **Results:** Patch size 256 yielded 29.05 dB PSNR, outperforming patch size 128 which scored 28.43 dB.

	e3: patch_size = 256	e4: patch_size = 128
PSNR	29.05	28.43

4. **Implications:** Larger patch sizes enable better restoration due to increased receptive field and context awareness.

5. Analysis

Why choose this architecture? Pros and Cons

The model I selected is the official implementation provided by the authors of the PromptIR paper, available in their open-source GitHub repository (`model.py`). I chose this model because:

- **Authenticity and Reliability:** Using the original author's code ensures faithful reproduction of the architecture and experimental settings, minimizing discrepancies that often arise from re-implementations.
- **State-of-the-Art Design:** The model leverages a hierarchical transformer backbone combined with prompt generation blocks, which together effectively handle multiple unknown degradations in a unified framework.
- **Efficiency and Flexibility:** The design balances high restoration performance with computational feasibility, making it suitable for training from scratch on the provided dataset.
- **Modular Prompt Conditioning:** The prompt blocks inject learned degradation-specific signals at multiple decoder stages, which is an elegant solution for all-in-one restoration without explicit degradation classification.

Pros:

- Handles multiple degradation types in one model without separate branches.
- Leverages transformers to capture long-range spatial dependencies.
- Prompt-based conditioning enhances adaptability and robustness.

Cons:

- Relatively large model size (~35 million parameters) requires significant GPU memory and training time.
- Complexity in architecture can make modifications and debugging more challenging.
- Dependency on carefully tuned training strategies (warm-up, EMA) to achieve stable convergence.

6. References

- Yi, X., Potlapalli, V., Zamir, S. W., Khan, S., & Khan, F. S. (2023). *Prompting for All-in-One Blind Image Restoration*. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). https://arxiv.org/abs/2306.13090
- Official PromptIR GitHub Repository: https://github.com/va1shn9v/PromptIR