

# 111550173\_HW4 Report

## Introduction.

This task was to restore images affected by two types of degradations: raindrops and snowflakes. At first, I tried to build my own model by connecting the encoder, decoder, and prompt blocks myself, but the results fell far short of expectations. So later, I switched to using the PromptIR[1] model made available on GitHub by the original paper as the foundation for training, and with that, I was able to complete the assignment.

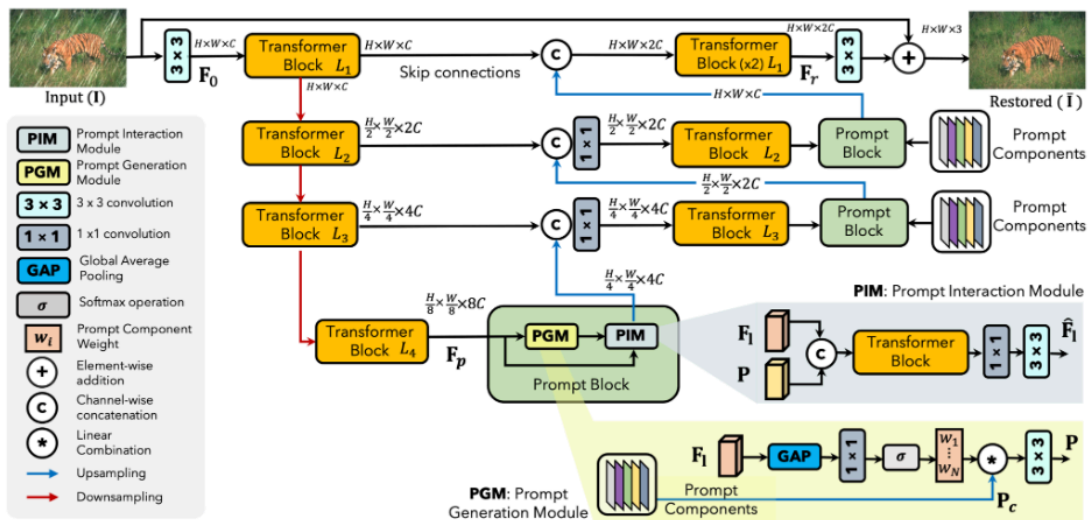
## Method.

The entire code is divided into the following sections:

- import
- model loading(pretrained weight is not allowed)
  - PromptIR: Prompting for All-in-One Blind Image Restoration
    - This architecture targets blind image restoration, meaning it restores images degraded by unknown factors (like blur, rain, noise, etc.) without needing prior knowledge of the degradation type.
  - Encoder Part
    - Input image → passes through a convolutional layer to get feature
    - Then goes through 4 Transformer Blocks[2]:
      - L1: keeps original resolution  $H \times W \times C$
      - L2: downsampled to  $H/2 \times W/2 \times 2C$
      - L3: further downsampled to  $H/4 \times W/4 \times 4C$
      - L4: deepest layer  $H/8 \times W/8 \times 8C$
  - Prompt Block Part
    - This is the core innovation of the model: using prompts to guide the restoration process across different feature levels.
    - PGM (Prompt Generation Module):
      - Generates prompt component  $P_c$  from the deepest feature  $F_p$ 
        - Apply Global Average Pooling to compress spatial dimensions.
        - Use softmax to determine weights  $w_i$  for each component.

- Perform a linear combination to produce the final prompt  $P$
- PIM (Prompt Interaction Module):
  - Integrates the prompt  $P$  with the encoder's intermediate features  $F_l$ 
    - Concatenate feature and prompt.
    - Pass through a transformer block.
    - Apply a  $3 \times 3$  convolution to output updated  $F_l^{\wedge}$
- Decoder Part
  - Combines skip connections and upsampling:
    - Gradually upsamples the deepest features.
    - Merges them with the corresponding encoder outputs + prompt outputs.
    - After each merge, it passes through a transformer block to strengthen the features.
    - Finally outputs the restored image  $I^{\wedge}$
- ALL the parameters remain the same with the code on PromptIR's github release.

## PromptIR: Prompting for All-in-One Blind Image Restoration



[Paper link](#)

- data process
  - random crop the origin image into  $128 \times 128$  (origin  $256 \times 256$ ), cropping into this size because a bigger size of model input causes a huge GPU memory used, a T4 has a 15GB space, and in this way will spend 13.4 GB, reaching the limit
  - random horizontal flip
  - random vertical flip

- implying the same process to both degraded data and clean data
  - training
    - LR rate: 1e-4
    - epoch: 50
    - optimizer: Adam
    - loss: L1loss
  - evaluate
    - use train data as the test data
    - calculate the PSNR with formula
- ```
mse = nn.functional.mse_loss(pred, target)
return 20 * torch.log10(max_pixel / torch.sqrt(mse + 1e-8))
```
- prediction
    - send test images into the model
    - turn the output into npz

## Results.

During training, I tried two types of data transforms applied to the training data. One was resizing the entire image to 128×128, and the other was the previously mentioned 128 random crop. Although the former showed very good PSNR scores on the validation set, due to a significant domain gap with the test data, the final scores were not ideal. After printing out the restored images for observation, I noticed that the restoration worked quite well on raindrop images, but the performance on snowflake images still needs improvement — there were still some visible white spots (in the figure below, 0 indicates snow, 1 indicates rain).

```
Epoch 28/30: 100%|██████████| 800/800 [15:55<00:00, 1.19s/it]
```

```
[Epoch 28] Loss: 0.0179, PSNR: 31.41
```

```
Epoch 29/30: 100%|██████████| 800/800 [15:55<00:00, 1.19s/it]
```

```
[Epoch 29] Loss: 0.0176, PSNR: 31.60
```

```
Epoch 30/30: 100%|██████████| 800/800 [15:54<00:00, 1.19s/it]
```

```
[Epoch 30] Loss: 0.0175, PSNR: 31.65
```

0.png



1.png



|        |          |                  |          |       |                                                                                                                                                                                                                                                             |
|--------|----------|------------------|----------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 288638 | pred.zip | 2025-05-14 22:44 | Finished | 23.89 |    |
|--------|----------|------------------|----------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

As a result, I eventually adopted the approach described in the Method section, avoiding compression of the original images, which led to the final version.

Although this method did not achieve outstanding PSNR performance during training, it still achieved a score close to the strong baseline on the scoreboard.

```
Epoch 39/40: 100% |██████████| 800/800 [16:32<00:00, 1.24s/it]

[Epoch 39] Loss: 0.0289, PSNR: 26.89

Epoch 40/40: 100% |██████████| 800/800 [16:27<00:00, 1.23s/it]

[Epoch 40] Loss: 0.0288, PSNR: 26.92
```

|        |          |                  |          |       |                                                                                                                                                                                                                                                                   |
|--------|----------|------------------|----------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 290065 | pred.zip | 2025-05-16 07:41 | Finished | 29.51 |    |
|--------|----------|------------------|----------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Reference.

[1] Vaishnav Potlapalli, Syed Waqas Zamir, Salman H Khan, and Fahad Shahbaz Khan. PromptIR: Prompting for all-in-one image restoration. Advances in Neural Information Processing Systems (NeurIPS), 36, 2024.

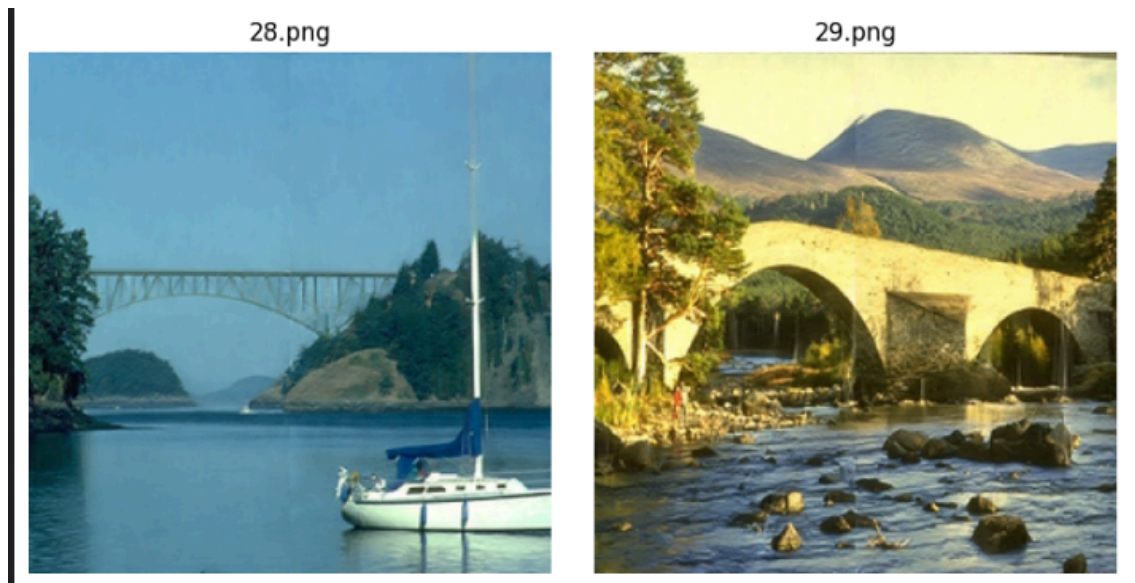
[2]

Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., & Wang, Y. (2021). Transformer in Transformer. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, & J. W.

Vaughan (Eds.), *Advances in neural information processing systems 34: annual conference on neural information processing systems, NeurIPS, December 6-14, virtual* (pp. 15908–15919).

### **Additional experiment.**

Since the model was trained using  $128 \times 128$  cropped patches as training data, I hypothesized that during testing, I could similarly split a  $256 \times 256$  image into four  $128 \times 128$  patches, perform predictions on each patch individually, and then stitch the predicted results back together. I thought this might lead to some unexpected improvements. However, as it turned out, while the results indeed changed, the reconstructed images exhibited clear stitching artifacts, and the overall performance score showed no improvement. It was ultimately an unsuccessful attempt.



In PromptIR, the prompt length controls how much additional guiding information (prompt tokens) the model injects to help adjust features and enhance restoration. This setting affects the model's flexibility and representational capacity when handling different degradation types. If the length is too short, it may not provide enough adjustment space; if it's too long, it can lead to wasted computational resources or even overfitting. Therefore, finding a suitable prompt length is a key design challenge.




Based on this consideration, I adjusted the model's prompt length to test whether it could achieve better prediction performance. Specifically, I varied the length from 3 to 5, training each configuration for 40 epochs. The experimental

results showed no significant differences in training loss or validation PSNR, but the model with a length of 5 performed better on the leaderboard.

Length = 3:

|        |            |                  |          |       |                                                                                                                                                                                                                                                             |
|--------|------------|------------------|----------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 299462 | pred_2.zip | 2025-05-28 02:14 | Finished | 28.89 |    |
|--------|------------|------------------|----------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Length = 4:

|        |            |                  |          |       |                                                                                                                                                                                                                                                             |
|--------|------------|------------------|----------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 293110 | pred_2.zip | 2025-05-19 23:44 | Finished | 29.28 |    |
|--------|------------|------------------|----------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Length = 5:

|        |          |                  |          |       |                                                                                                                                                                         |
|--------|----------|------------------|----------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 290065 | pred.zip | 2025-05-16 07:41 | Finished | 29.51 |   |
|--------|----------|------------------|----------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Github Link.**

<https://github.com/Seanlin0911/NYCU-DLCV-2025-Spring/tree/main>