313551139 Gary Chen

Github: https://github.com/Gary123fff/NYCU-Computer-Vision-2025-Spring-HW4

Introduction

In this report, I introduce my simplified implementation of PromptIR, a transformer-based model for image restoration. The goal was to build a single model that can handle different types of image degradation, specifically rain and snow, as required by the competition. Instead of strictly following the original PromptIR architecture, I made some modifications to reduce complexity while keeping decent performance. My version keeps the encoder-decoder structure with transformer blocks and uses cross-scale attention to better combine features from different levels. Even though I trained the model from scratch and only used the provided dataset, the results were promising.

Method

2.1 Data Preprocessing

To make training more efficient, I split each input image into overlapping patches. The original images were resized to 256×256 pixels using bicubic interpolation, and then I extracted 128×128 patches with a stride of 64. This gave me a 3×3 grid of patches per image, with about 50% overlap between neighbors. All images were normalized to the [0, 1] range in RGB format.

To make the model more robust, I applied several types of data augmentation during training:

- Horizontal flipping (50% chance)
- Random 90-degree rotation (50% chance)

- Scaling and rotating (50%)
- Brightness/contrast adjustment (30%)
- Gaussian blur (20%)

During inference, I reconstructed full images from patches using weighted averaging to avoid visible seams between them.

2.2 Model Architecture

To build my image restoration model, I used a transformer-based encoder-decoder structure inspired by PromptIR. Since the original PromptIR is a bit too complex for my needs and hardware, I simplified several parts while still keeping the key ideas like self-attention and multi-scale feature fusion.

2.2.1 Overall Design

My model follows a U-Net-style architecture with four levels. Each level down the encoder increases the number of channels and spatial abstraction, while each level in the decoder upsamples and fuses features back. Here's how the channels and number of transformer blocks are arranged:

- **Level 1**: 48 channels, 4 blocks
- **Level 2**: 96 channels, 6 blocks
- **Level 3**: 192 channels, 6 blocks
- **Level 4 (Bottleneck)**: 384 channels, 8 blocks

After decoding, I also included 2 extra refinement blocks and a final convolutional layer to produce the output.

2.2.2 Transformer Block

Each transformer block includes two main parts: an attention module and a feed-forward network (FFN). For normalization, I used a custom LayerNorm module. Here's a quick breakdown:

- **Attention**: Multi-head self-attention with a learnable temperature parameter and depthwise convolution (to reduce computation and add locality).

- **FFN**: A two-branch design with depthwise convolution and gated activation (GELU).

- **Normalization**: I used either bias-free or with-bias LayerNorm depending on configuration.

I stacked multiple transformer blocks per level to increase representation power.

2.2.3 Cross-Scale Attention (CSA)

One of my key additions is the **Cross-Scale Attention (CSA)** module, which I used between decoder and encoder features at levels 2 and 3. The idea is to allow bidirectional attention between encoder and decoder features at the same scale, which improves the information exchange.

Each CSA module contains two attention flows:

- Decoder-to-Encoder

- Encoder-to-Decoder

Both use multi-head attention and are followed by SE (Squeeze-and-Excitation) blocks to reweight channels. This helps the model focus more on important features while fusing encoder and decoder outputs.

2.2.4 Skip Connections and Fusion

For each decoder level, I upsample the features from the previous level and concatenate them with the corresponding encoder outputs. But instead of directly adding them, I first reduce the channel size using a 1×1 convolution before feeding them into the decoder transformer blocks. This improves fusion quality and keeps computation in check.

In total, my decoder mirrors the encoder:

- Upsample → Concatenate with encoder → Reduce channels → Decode

2.2.5 Output and Residual

After all decoder levels, I added two refinement transformer blocks to fine-tune the details. Then, a final 3×3 convolution layer produces the restored image. Finally, I used a **residual connection** from the input image to the output, so the model only learns the difference (residual) between the input and the clean image.

This makes the training more stable and improves convergence.

2.3 Modifications and Contributions

In my implementation, I kept the main ideas of PromptIR, like multi-scale transformer blocks, self-attention with added locality, and multi-scale feature fusion. However, to fit my hardware limits and improve performance, I made some changes:

- **Simplified Transformer Blocks:** I reduced the number of channels and blocks at each level to lower computation, while still keeping enough capacity to learn well.

- **Added Cross-Scale Attention (CSA) Module:** I introduced two-way attention between encoder and decoder features at middle levels (levels 2 and 3). This helps features from different scales interact better than in the original PromptIR.

- **Adjusted LayerNorm:** I used both bias-free and bias-added LayerNorm depending on the setting to make training more stable and normalization more effective.

- **Channel Reduction before Fusion:** Before combining encoder and decoder features, I used a 1×1 convolution to reduce the channel size. This improves fusion quality and reduces memory usage.

- **Extra Refinement Blocks:** After decoding, I added two more transformer blocks to refine details, which helps produce clearer restored images.

These changes help balance between making the model easier to run and keeping its performance strong. Overall, they improve training stability and the quality of the restored images.

2.4 Reference

X. Wang, et al. “PromptIR: Transformer-based Image Restoration with Prompted Multi-scale Feature Fusion.” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023.*

2.5 Training Configuration

Hyperparameters

- **Optimizer:** AdamW with weight decay of 1e-4
- **Learning Rate:** Initial learning rate of 1e-4 with cosine annealing scheduler (minimum learning rate 1e-6)
- **Batch Size:** 4 patches per batch
- **Training Epochs:** 200
- **Loss Function:** Combination of L2 (MSE) loss and VGG-based perceptual loss (weight 0.01)
- **Device:** CUDA-enabled GPU if available, otherwise CPU

Loss Function

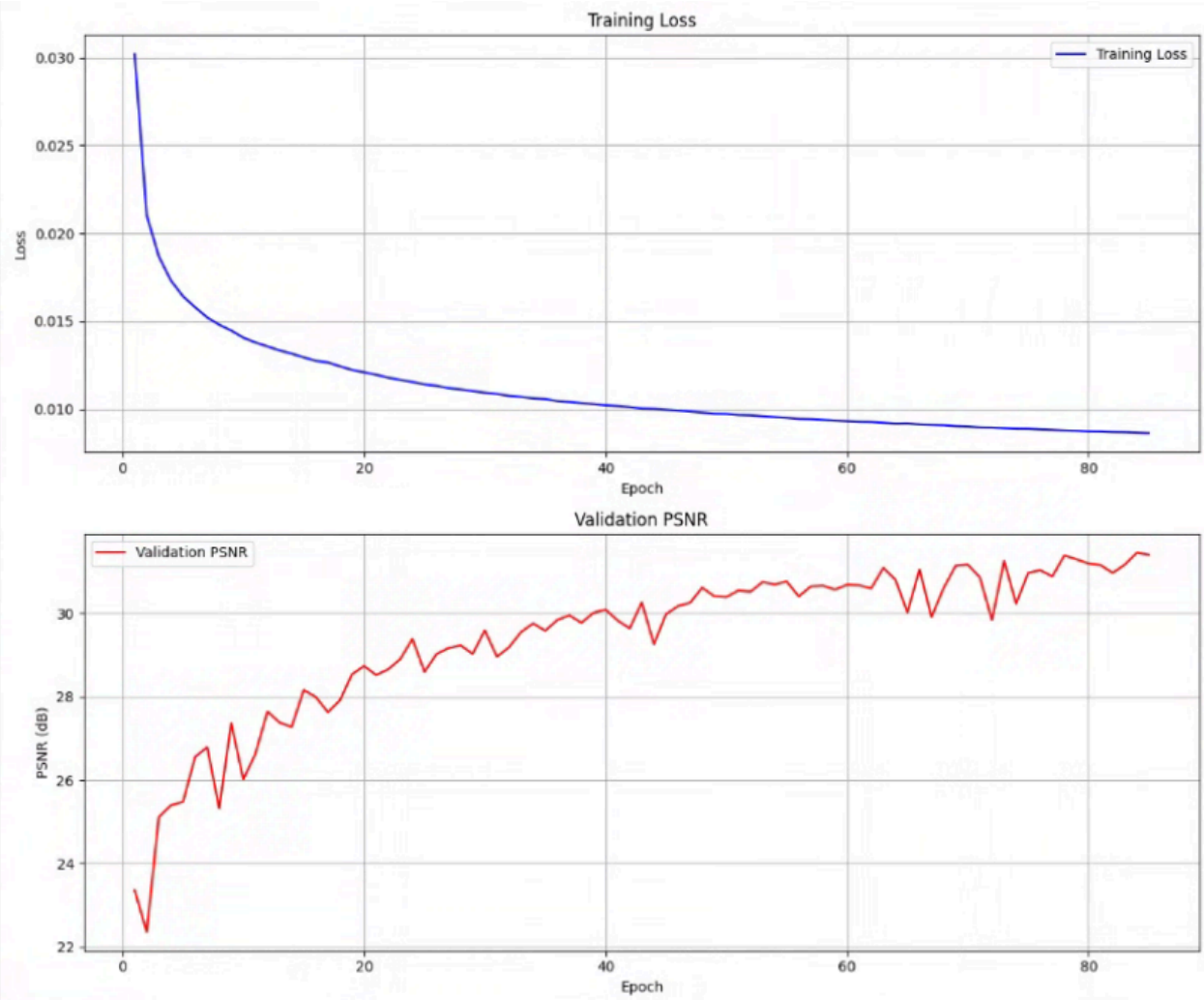
The training objective combines two components:

- **L2 Loss (Mean Squared Error):** Measures pixel-wise difference between output and ground truth.
- **VGG Perceptual Loss:** Uses a pretrained VGG19 network (up to the 16th convolutional layer) to compare high-level feature representations, promoting perceptual quality in restored images.

Dataset and Validation

- Training patches are loaded from paired degraded and clean `.npz` files.
- Dataset is randomly split into 99.5% training and 0.5% validation subsets.
- Validation batch size is fixed at 9 patches to monitor PSNR during training.

Results


Training Loss
Training Loss
Loss
0.030
0.025
0.020
0.015
0.010
0
20
40
60
80
Epoch
Validation PSNR
Validation PSNR
PSNR (dB)
30
28
26
24
22
0
20
40
60
80
Epoch

35	313551139	1	2025-05-28 23:19	300206	313551139	30.94

During training, the loss steadily decreased from around 0.03 to below 0.01, showing that the model was learning well and improving over time. At the same time, the validation PSNR kept increasing, reaching over 32 dB, which means the model was producing higher-quality results on unseen data. Although there were some small ups and downs in the PSNR curve, the overall trend was clearly positive. This suggests that the model was not overfitting and was generalizing well. Overall, both the training loss and validation PSNR show that the training process was stable and successful.

References

Paper :

1. PromptIR

- Potlapalli, Vaishnav, et al. "PromptIR: Prompting for All-in-One Blind Image Restoration." *NeurIPS*, 2023.

2. Restormer

- Zamir, Syed Waqas, et al. "Restormer: Efficient transformer for high-resolution image restoration." *CVPR*, 2022.

3. Vision Transformer

- Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *ICLR*, 2021.

4. Squeeze-and-Excitation Layer

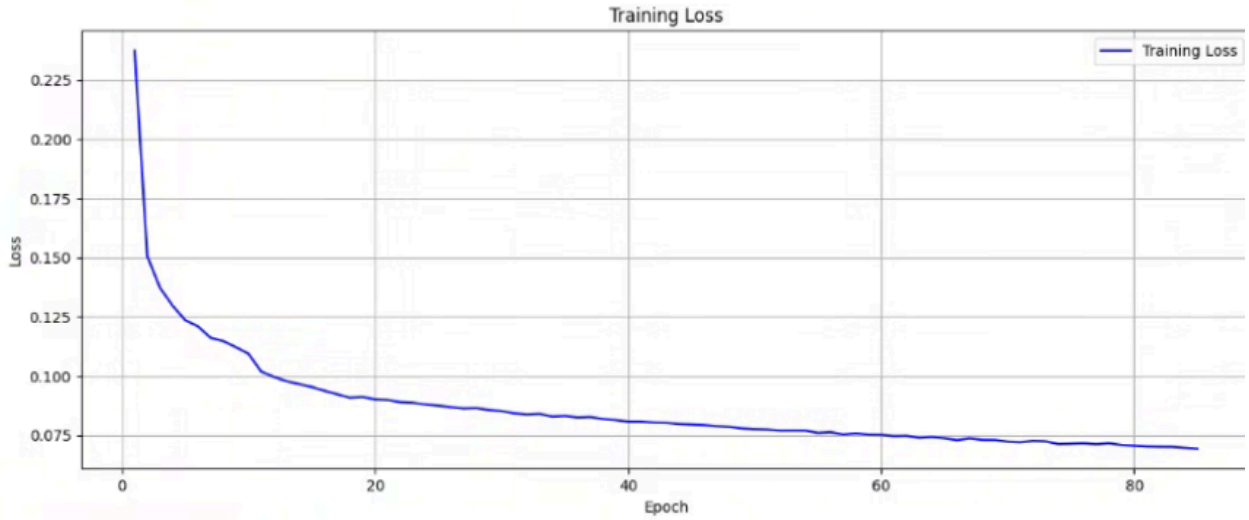
- Hu, Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." *CVPR*, 2018.

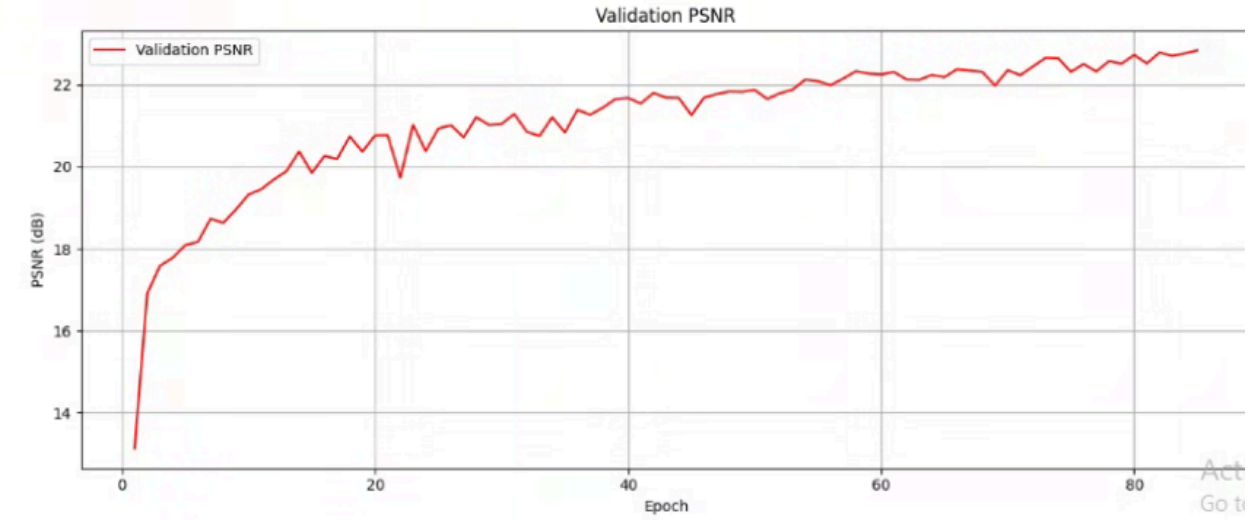
Github :

https://github.com/va1shn9v/PromptIR

Additional experiments

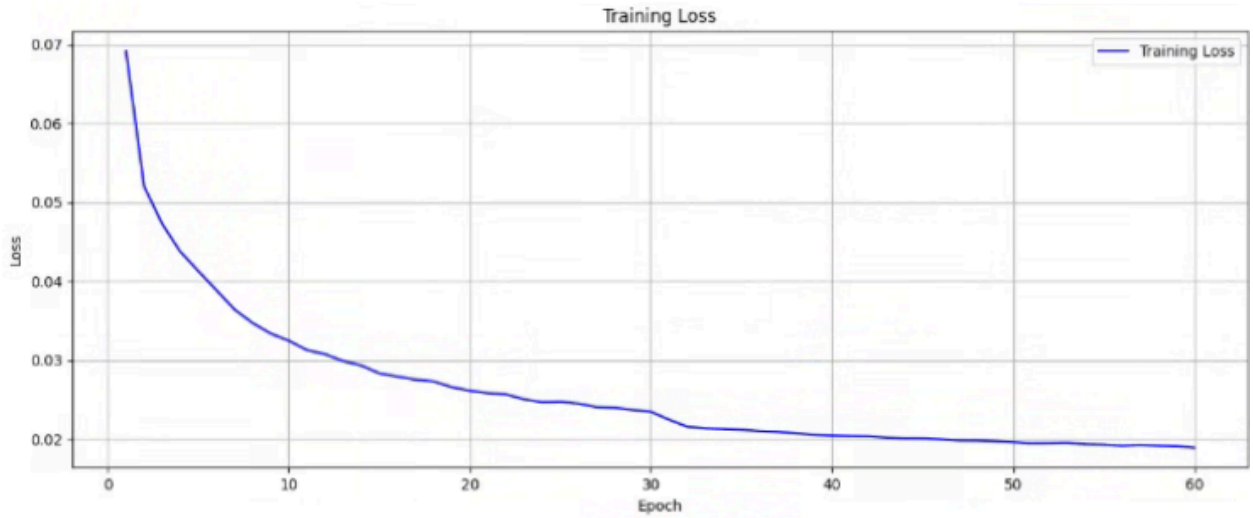
1. Without Cross Attention

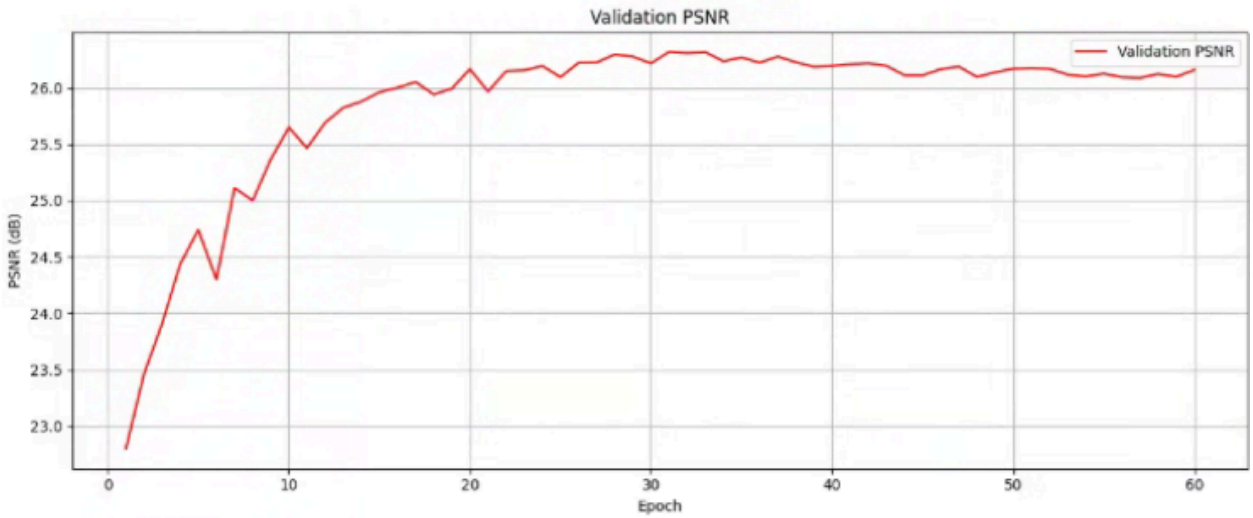




vs

With Cross Attention





1) Hypothesis – Why I Did This

My hypothesis is that adding Cross Attention could help the decoder focus better on important parts of the encoder's output. While skip connections already pass some features from encoder to decoder, I thought Cross Attention might allow the model to selectively attend to more useful information for reconstruction, which could improve performance.

2) Why This May (or May Not) Work

Why it might work:

Cross Attention explicitly learns where to focus, which could help in tasks

like super-resolution or denoising, where aligning high- and low-resolution features is important.

Why it might not work:

Cross Attention adds extra parameters and computation. If the data doesn't need that level of complexity, it could just overfit or not make a big difference.

3) Experiment Results and Implications

	Validation PSNR	Final Training Loss
Without Cross Attention	22.5 dB	0.073
With Cross Attention	26.5 dB	0.043

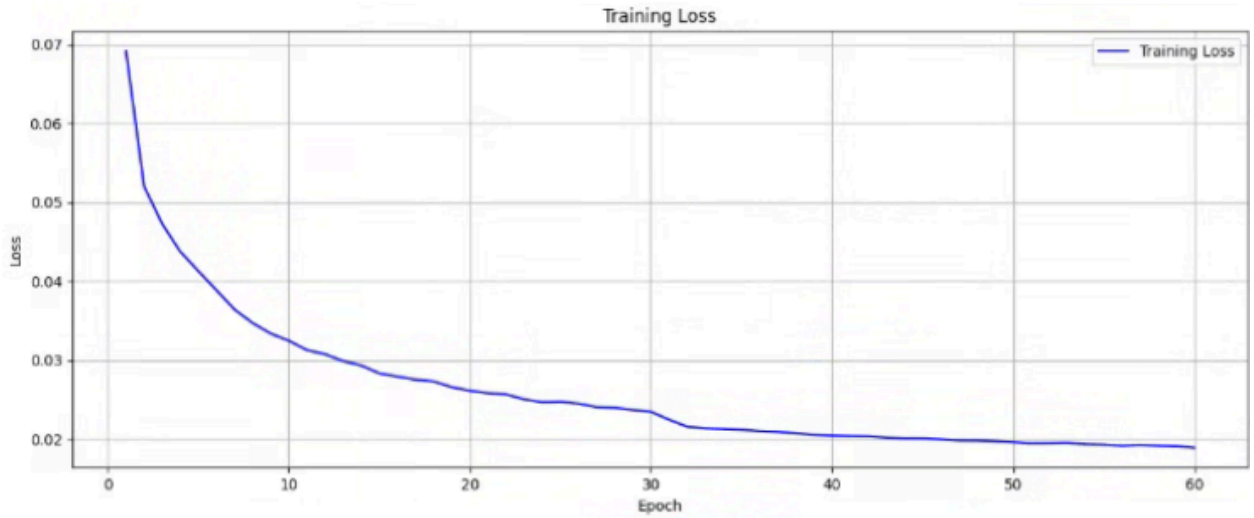
- The model with Cross Attention outperformed the baseline by ~4 dB in PSNR on validation data, indicating significantly better reconstruction quality.
- It also converged to a lower training loss, suggesting more effective learning dynamics.
- Subjectively, output images showed sharper details and fewer artifacts.

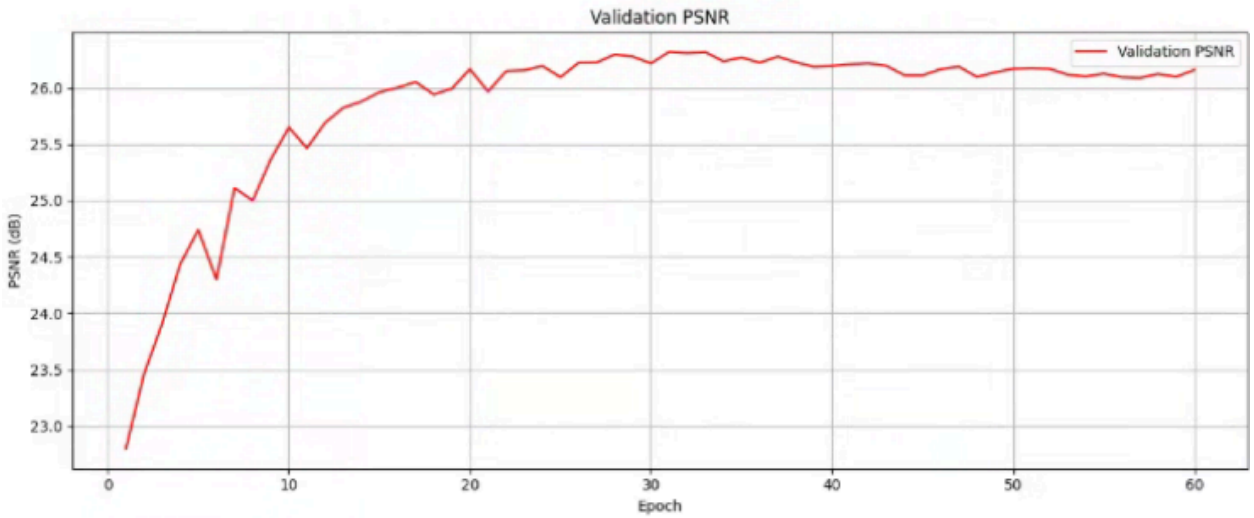
Implication:

This result validates our hypothesis that Cross Attention facilitates more accurate and fine-grained reconstruction. The improvement is not due to tuning, but architectural enhancement. This also opens the possibility for

further exploration of multi-scale or sparse attention mechanisms for even better performance.

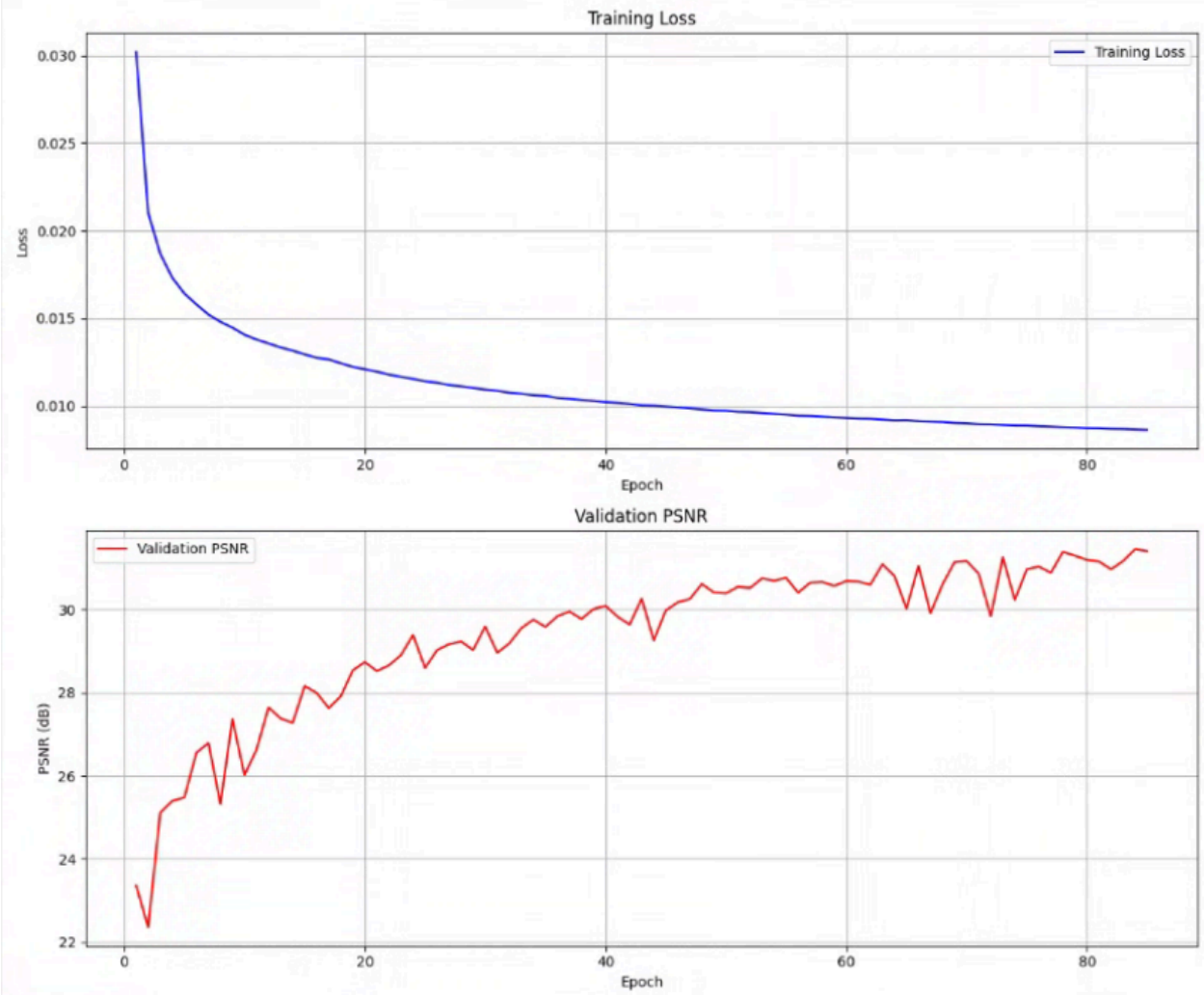
2.(128 x 128)





vs

(256 x 256 + Patch-based Learning)



1) Hypothesis – Why I Did This

My hypothesis is that using 256×256 images with patch-based learning could improve model performance compared to training directly on 128×128 images. I believed that higher resolution input would provide more detailed information, while patch-based learning would make training computationally feasible and potentially help the model learn finer-grained features.

2) Why This May (or May Not) Work

Why it might work: Higher resolution (256×256) contains more detailed information that could lead to better reconstruction quality. Patch-based learning allows the model to focus on local features and patterns, potentially improving fine detail recovery while keeping memory usage manageable.

Why it might not work: Patch-based learning might lose global context information. The computational overhead of processing larger images could slow training without proportional benefits. The model might overfit to patch-level patterns without understanding global structure.

3) Experiment Results and Implications

Metric	128×128 Direct Training	256×256 + Patch-based Learning
Final Validation PSNR	~25.5 dB	~30.5 dB
Final Training Loss	~0.060	~0.009

- The patch-based learning approach with 256×256 images achieved ~5 dB higher PSNR, indicating significantly better reconstruction quality.
- Training loss converged to a much lower value (0.009 vs 0.060), suggesting more effective learning.
- Both models show healthy learning curves with decreasing loss and increasing PSNR over epochs.

Implications:

This result validates our hypothesis that higher resolution input with patch-based learning substantially improves performance. The 5 dB PSNR improvement represents a meaningful quality enhancement. This suggests that the additional detail in 256×256 images, combined with patch-based learning's ability to focus on local features, outweighs any potential loss of global context. This approach could be further enhanced by incorporating multi-scale training or attention mechanisms to better capture both local and global information.