Visual Recognitionusing Deep Learning
2025 Spring, Homework 4

313551078 吳年茵

Github：`https://github.com/nianyinwu/CV_HW4`

1. Introduction

In this lab, we have to train an image restoration model to solve an image restoration task to let the two types of degraded images, Rain and Snow, be clean and design other tricks to improve the model's performance, such as modifying the architecture. We use a dataset of 1600 degraded images per type for training and validation and 50 for testing, with a resolution of 256 × 256.

As the base model, I adopted PromptIR [1], a unified image restoration framework that dynamically uses prompt-based modulation to adapt to different degradation types. Instead of training task-specific models, PromptIR uses a shared encoder-decoder architecture enhanced with prompt blocks. Each prompt block includes a Prompt Generation Module (PGM) and a Prompt Interaction Module (PIM). The PGM generates degradation-aware embeddings, and the PIM modulates decoder features accordingly to guide restoration.

I used the PromptIR from the official public GitHub repo [2] and trained from scratch using no pre-trained weights to match the lab constraints and focus on derain and desnow tasks. Additionally, I experimented with some tricks of enhancements such as integrating attention modules and adjusting the number of decoder blocks to improve PSNR (Peak Signal-to-Noise Ratio), which measures restoration quality based on mean squared error.

2. Method

A. Data Preprocessing

In the data preprocessing, I split the training data into training and validation with the ratio of 8:2. I used different resolutions to experiment and train the model for this task and also did some data preprocessing and augmentation techniques to the data using PyTorch's transforms module [3].

The preprocessing process of training data is as follows:

1. Convert the image into a PyTorch tensor.
2. Randomly adjust image sharpness with a factor of 2.0, applied with a probability of 0.4.
3. Randomly flip the image horizontally with a probability of 0.4.
4. Randomly flip the image vertically with a probability of 0.4.
5. Randomly apply an affine transformation including rotation within ±5°, translation up to 5%, and scaling from 95% to 105% with a probability of 0.4.
6. Convert the tensor to float32 and scale pixel values from [0, 255] to [0.0, 1.0].

The preprocessing process of validation and testing data is as follows:

1. Convert the image to a PyTorch tensor.
2. Convert the tensor to float32 and scale pixel values from [0, 255] to [0.0, 1.0].

B. Model Architecture

B.1. Base Architecture

The architecture of PromptIR uses a UNet-style network [4] with transformer blocks in the encoding and decoding stages and enhanced by prompt-based modulation. The model is composed of the following main components:

1. Encoder
   The input image will first be passed through a 3×3 convolutional layer to extract initial features. These features are then processed through a series of Transformer blocks, with progressive downsampling, so the encoder will gradually capture multi-scale representations with increasing feature channels.

2. Prompt Block

   At the bottleneck and decoder stages, it will integrate Prompt Blocks, each consisting of two components:

   (a) Prompt Generation Module (PGM)

       This module dynamically generates input-aware prompts to guide the restoration process. Instead of using fixed prompts, it computes attention weights from input features using global average pooling and a softmax function, and applies them to a set of learnable prompt components. This produces adaptive prompts tailored to the specific degradation characteristics of each input, enabling the model to better generalize across diverse restoration tasks.

   (b) Prompt Interaction Module (PIM)

       The Prompt Interaction Module modulates decoder features using input-conditioned prompts from PGM. It integrates prompt embeddings via channel-wise concatenation, followed by a Transformer block and convolution layers to refine features in a degradation-aware manner. This allows the model to selectively enhance or suppress feature responses based on the degradation type, thereby improving restoration quality.

3. Decoder

   In the decoder stage, features are progressively upsampled and fused with corresponding encoder outputs through skip connections. Each stage includes Transformer blocks to further refine the features. Finally, a 3×3 convolution is applied, and the result is element-wise added to the initial shallow features to produce the restored image.

B.2. Modification (including additional experiments modification)

In this section, I will present the modifications in the PromptIR architecture to gain better PSNR metrics in the testing phase:

(1) Introduce a Convolutional Block Attention Module (CBAM) [5] after the encoder. CBAM applies channel and spatial attention sequentially, encouraging the model to focus on more informative and task-relevant regions in the feature maps. This addition is expected to enhance the model's capacity to emphasize meaningful content before entering the decoding stage. In addition, I used a combined loss function for training, consisting of L1 loss and Structural Similarity (SSIM) [6] loss. The L1 loss enforces pixel-wise

accuracy, while the SSIM loss encourages perceptual similarity between the restored and ground-truth images, helping the model to produce visually more faithful results.

(2) In the original PromptIR design, the number of decoder blocks from deep to shallow layers was configured as 8, 6, and 6. To enhance the model's decoding capacity and improve restoration quality, I modified the configuration to 8, 8, and 8, aiming for more balanced and deeper refinement across all levels.

(3) I also found that the number of refinement blocks in the last Transformer decoder block was only 4, which seemed relatively shallow. I suspected this might limit the model's ability to effectively restore fine details in the output images. Therefore, I conducted an additional experiment by increasing the number of refinement blocks to 8, hoping that a deeper decoder would enable the model to capture complex features better and improve the overall restoration quality.

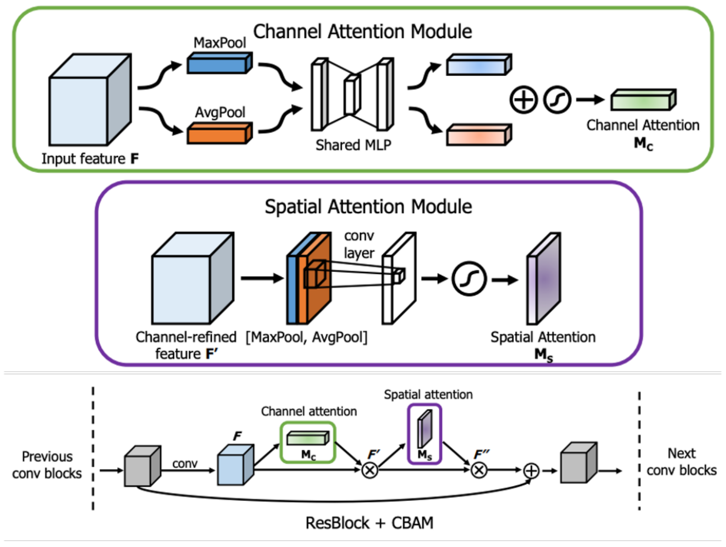


Fig. 1. CBAM module architecture from [5]

C. Hyperparameters Setting

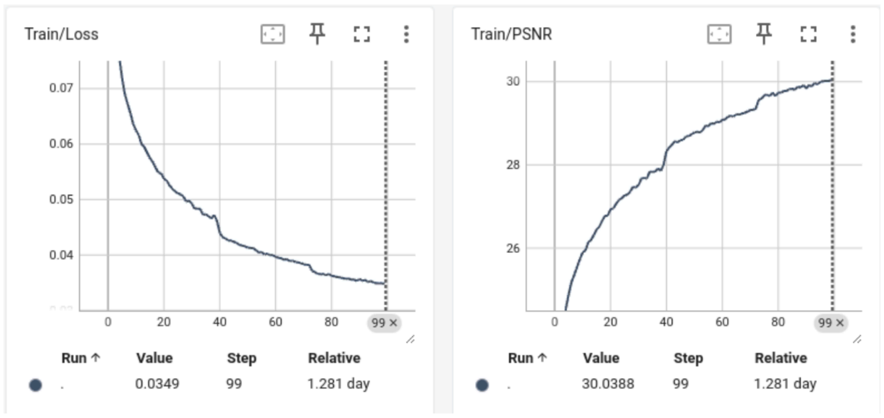
In the experiments, I used the following hyperparameter settings to train the model:

- Learning rate: 2e-4
- Batch size: 4
- Number of epochs: 100
- Optimizer: AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 5×10^{-4}.
- Learning rate scheduler: Use ReduceLROnPlateau to monitor the validation loss during training. If loss doesn't decrease for 5 consecutive validations, the learning rate is reduced by a factor of 0.5, with a minimum learning rate set to 1×10^{-6}.

3. Experiment Results

I conducted four experiments to explore better performance: (1) PromptIR with L1 Loss and SSIM Loss (Baseline) (2) Baseline that introduces the CBAM module (3) Modify (2) model's decoder blocks number to fine-tune with epoch 20 (4) Based on (3) modified the number of refinement blocks and train from scratch

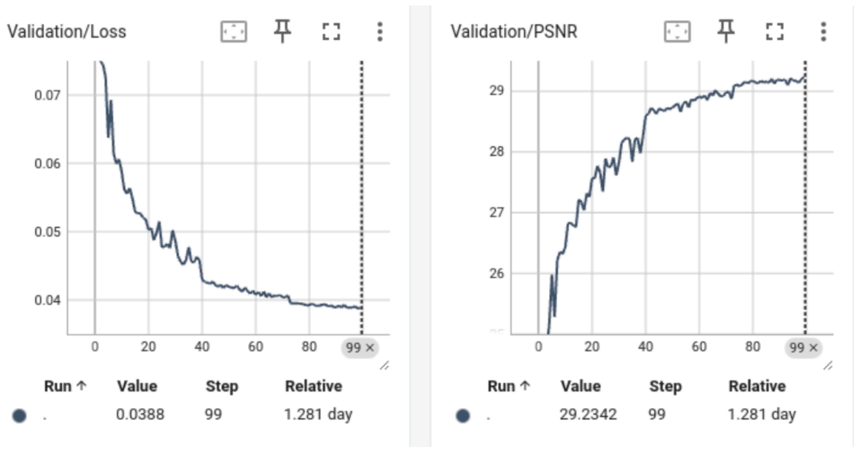
1. PromptIR with dual-loss training objective (Baseline)



Run ↑	Value	Step	Relative
● .	0.0349	99	1.281 day

Run ↑	Value	Step	Relative
● .	30.0388	99	1.281 day

(a) Training Curve



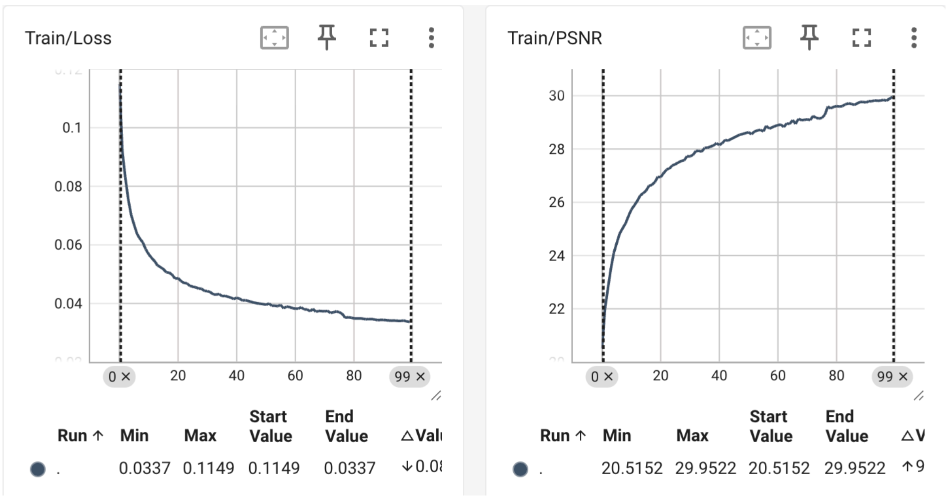
Run ↑	Value	Step	Relative
● .	0.0388	99	1.281 day

Run ↑	Value	Step	Relative
● .	29.2342	99	1.281 day

(b) Validation Curve

In the first experiment, we trained the baseline PromptIR model using a dual-loss training objective. As shown in the curves, training and validation losses reach approximately **0.035** and **0.039**. However, the training loss exhibits a relatively slow convergence in the earlier stages, suggesting that the optimization process may benefit from a better learning rate schedule or architectural refinement. Meanwhile, PSNR consistently improves, with the training reaching around 30.04 dB and the validation reaching 29.23 dB. These results indicate that the model achieves good reconstruction quality and generalization capability, but there is room for improvement in accelerating convergence and further performance.

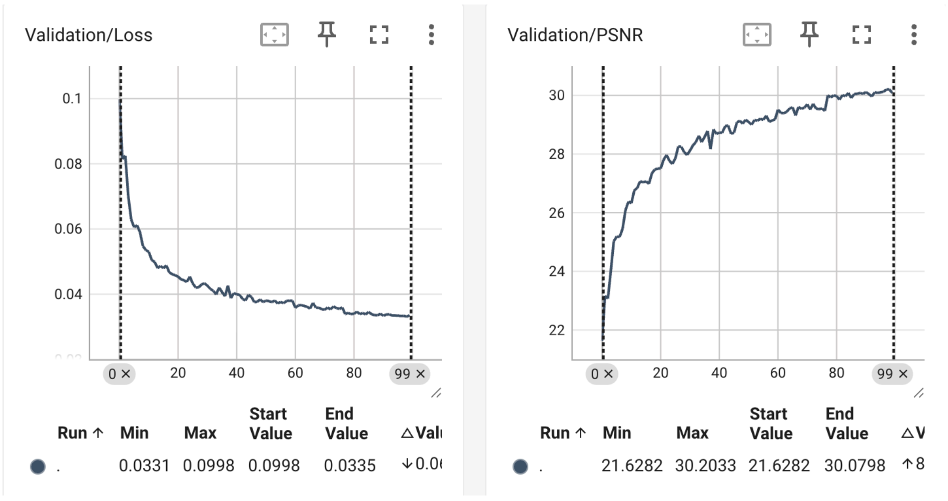
2. Baseline with CBAM module



Run ↑	Min	Max	Start Value	End Value	△Val
● .	0.0337	0.1149	0.1149	0.0337	↓0.0

Run ↑	Min	Max	Start Value	End Value	△V
● .	20.5152	29.9522	20.5152	29.9522	↑9

(a) Training Curve



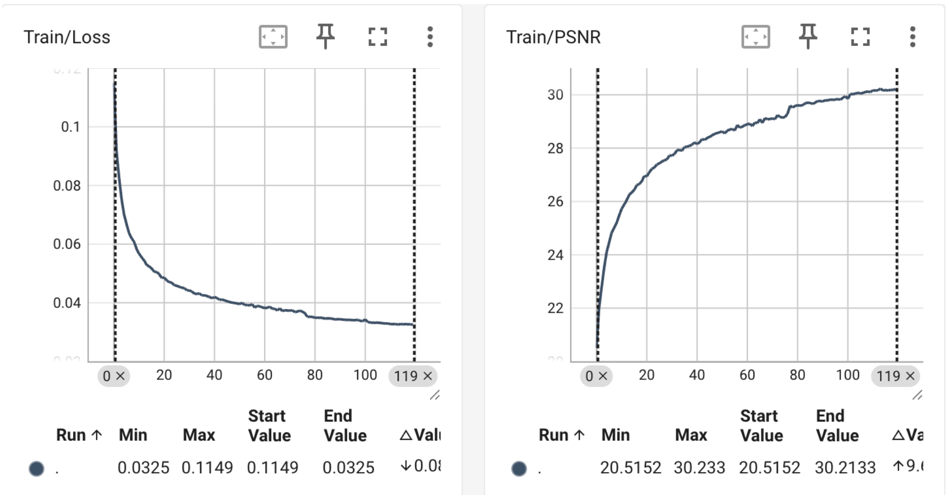
Run ↑	Min	Max	Start Value	End Value	△Val
● .	0.0331	0.0998	0.0998	0.0335	↓0.0

Run ↑	Min	Max	Start Value	End Value	△V
● .	21.6282	30.2033	21.6282	30.0798	↑8

(b) Validation Curve

In the second experiment, we integrated the CBAM module into the baseline architecture to enhance its spatial and channel-wise feature representation. The results show slight improvements in training and validation loss, with final values of **0.0337** and **0.0331**, respectively. However, while the training PSNR is slightly lower than the baseline, the validation PSNR significantly improves, reaching **30.2 dB**, surpassing the baseline' s 29.23 dB. In addition, this model converges faster than the baseline, particularly in the early stages of training, indicating that the attention mechanism helps focus more effectively on important regions. These results demonstrate that incorporating CBAM leads to better generalization and improved performance, especially on unseen validation data.

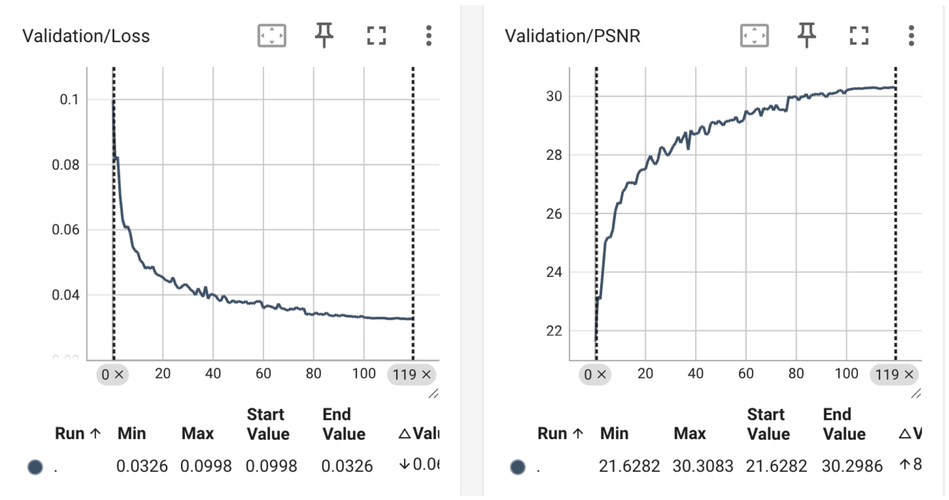
3. Modify (2) model's decoder blocks number to fine-tune with epoch 20



Run ↑	Min	Max	Start Value	End Value	△Val
● .	0.0325	0.1149	0.1149	0.0325	↓0.0

Run ↑	Min	Max	Start Value	End Value	△Va
● .	20.5152	30.233	20.5152	30.2133	↑9.

(a) Training Curve



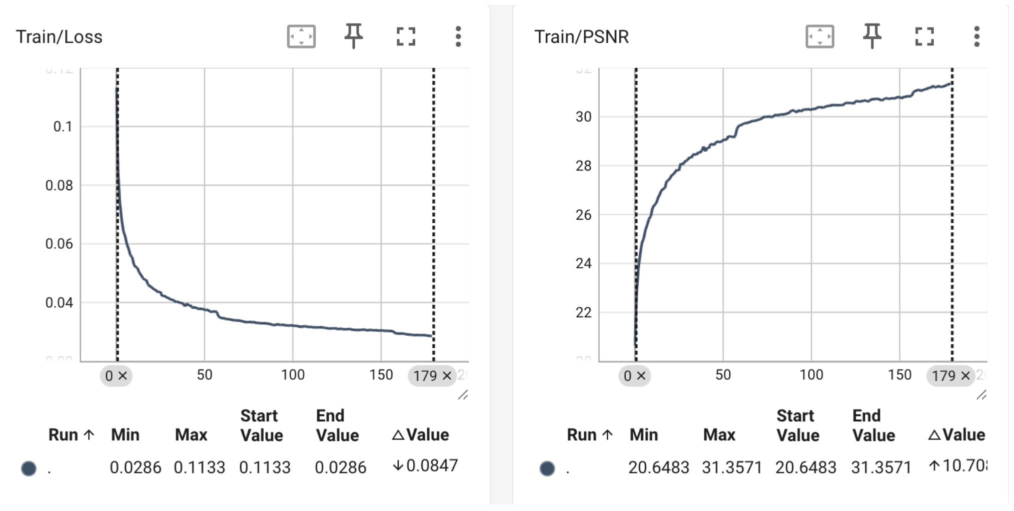
Run ↑	Min	Max	Start Value	End Value	△Val
● .	0.0326	0.0998	0.0998	0.0326	↓0.0

Run ↑	Min	Max	Start Value	End Value	△V
● .	21.6282	30.3083	21.6282	30.2986	↑8

(b) Validation Curve

In the third experiment, we modified the second model by increasing the number of decoder blocks to enhance its capacity to capture and reconstruct more detailed image features. The model was fine-tuned for 20 additional epochs using the best weights from the second experiment. As shown in the combined curves, training and validation losses further decreased to **0.0325** and **0.0326**, the lowest among all settings. The validation PSNR also improved to **30.31 dB**, surpassing the previous model. The PSNR curve remains smooth and steadily increasing, indicating that the added decoder depth improves reconstruction quality and generalization. These results confirm that increasing decoder blocks enhances the model's representational ability and overall performance.

4. Based on (3) modified the number of refinement blocks and train from scratch



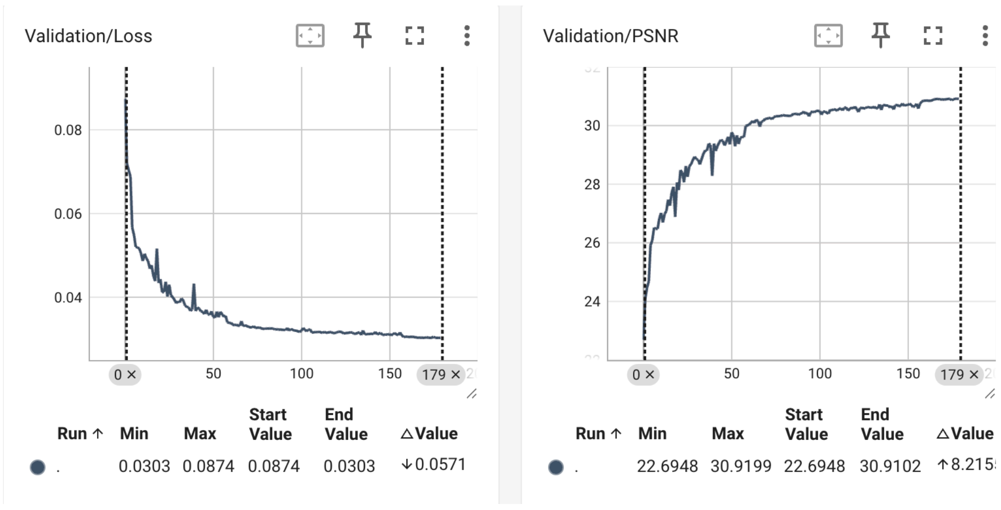
Train/Loss

Run ↑	Min	Max	Start Value	End Value	△Value
● .	0.0286	0.1133	0.1133	0.0286	↓0.0847

Train/PSNR

Run ↑	Min	Max	Start Value	End Value	△Value
● .	20.6483	31.3571	20.6483	31.3571	↑10.70

(a) Training Curve



Validation/Loss

Run ↑	Min	Max	Start Value	End Value	△Value
● .	0.0303	0.0874	0.0874	0.0303	↓0.0571

Validation/PSNR

Run ↑	Min	Max	Start Value	End Value	△Value
● .	22.6948	30.9199	22.6948	30.9102	↑8.215

(b) Validation Curve

In the final experiment, we extended the third model by increasing the number of refinement blocks and training the entire network from scratch. Based on observations from the previous experiments, we noticed that the PSNR continued to improve steadily, indicating that the model had not yet converged. Therefore, we trained the model for a longer duration, allowing sufficient epochs for it to learn and adapt fully—this modification aimed to enhance the model's ability to refine feature representations and produce higher-quality restorations iteratively. The curves show that training and validation losses decreased throughout the 180 epochs, reaching **0.0286** and **0.0303**, respectively—the lowest observed across all experiments. The validation PSNR also increased to **30.91 dB**, representing **the best performance among all previous experiments**. Notably, the PSNR curve exhibits a consistent upward trend with no signs of saturation, suggesting that the model is still

improving and could benefit from further training beyond 180 epochs. These results confirm the effectiveness of deeper refinement stages and highlight the potential of extended training in achieving better generalization and restoration quality.
Thus, for the Codabench submission, we selected the best performance checkpoint from the fourth model trained from scratch with a deeper refinement backbone. This final checkpoint achieved the best public PSNR score of **31.54 dB**, demonstrating the effectiveness of the extended architecture and training strategy.

References

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

[2] Vaishnav Potlapalli, Syed Waqas Zamir, Salman Khan, and Fahad Khan. Promptir: Prompting for all-in-one image restoration. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[3] http://pytorch.org/vision/main/transforms.html.

[4] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration, 2022.

[5] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module, 2018.

[6] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.