Empowering Resampling Operation for Ultra-High-Definition Image Enhancement with Model-Aware Guidance Supplementary Material

Anonymous CVPR submission

Paper ID 3204

1. Dataset Details

In this work, we validate the effectiveness of our methods using two datasets for UHD image enhancement. The UHD-LOL4K [7] dataset consists of 5999 paired training images and another 2100 paired test images. The 4KIL [4] dataset contains 1040 paired images, where we randomly select 940 images as the training dataset and the remaining 100 images are used for testing.

2. More Implementation Details

2.1. Pre-training Details

Our object is to establish the collaborative interplay between the resizers and existing enhancement models. We begin by training baseline models on the above two datasets. For all the baselines, we utilize the commonly adopted encoder-decoder architecture with skip connections, incorporating two times downsampling and upsampling within the backbone, as shown in Figure 1. The downsampling is achieved using a convolution layer with a stride of 2, while the upsampling is implemented using a transposed convolution layer. The pre-training process is based on the PyTorch framework with one NVIDIA 3090 GPU. To demonstrate the scalability and robustness of our methods, we pre-train three different backbones, including the CNN-I model on the UHD-LOL4K dataset, the CNN-H model on the 4KIL dataset, and the Restormer model on the 4KIL dataset. For the CNN-I model, the inner processing module is the invertible block proposed by [5]. For the CNN-H model, the inner processing module is the half instance normalization block proposed by [1]. The channel number is set to 64 for all the processing modules. For the Restormer model, we follow the default modules as proposed by [8] but with small channel numbers. Specifically, the number of transformer blocks is $[2, 3, 3]$, the number of heads is $[1, 2, 4]$, the number of channels is $[16, 32, 64]$. During pre-training, we adopted the Adam [3] optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for parameter optimization. For these two CNN models, we crop patch size of 1024×1024 for training and the batch size is set to 4. The training epochs are 100 for CNN-I and 30 for CNN-H. The initial learning rate is set to $1e^{-3}$ and $6e^{-4}$ for the CNN-I and CNN-H respectively, which decays by a factor value of 0.75 every 20 epochs and 10 epochs correspondingly. With regard to the Restormer backbone, we use a patch size of 768×768 and a batch size of 1 for training. The total training epoch is set to 30, and the initial learning rate is $5e^{-4}$ with a decay of 0.75 every 10 epochs. The mean absolute error is used for pre-training.

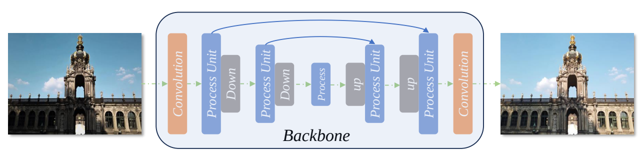

Figure 1. The architecture of the CNN backbone. The inside processing unit can be any module, where we utilize the invertible block and half instance block in our implementations.

2.2. Collaborative Training Details

In this stage, we aim to facilitate the collaboration between the resizer and enhancer via customizing resampling under the guidance of model knowledge. The overview of this process is depicted in Figure 2. It is important to note that the trainable parameters only consist of two convolution layers and one MLP, where the MLP is accelerated by tiny-cuda-nn [6]. The convolution layers are equipped with 1×1 kernels, a stride of 1, and no padding. For the CNN-I model, the MLP contains five hidden layers, each containing 128 neurons, and employing LeakyRelu as the activation function. For the CNN-H model, the MLP contains six hidden layers, each containing 128 neurons, and employing LeakyRelu as the activation function. For the Restormer, the MLP contains five hidden layers, each containing 128 neurons, and employing Tanh as the activation function. For the discriminator, the inner basic processing unit is the convolution layers followed by the LeakyRelu activation function. The convolution layers are equipped with 4×4 kernels, a stride of 2, and a padding size of 1.

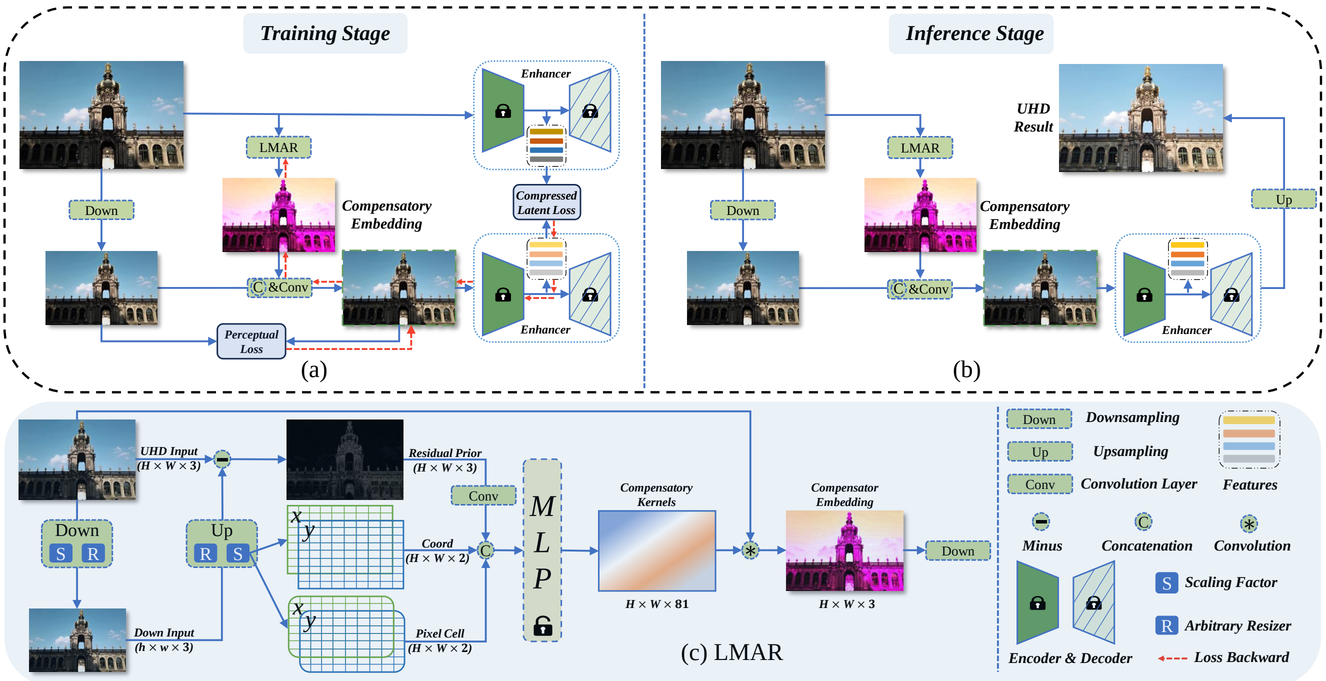

Figure 2. Overview of the proposed LMAR. The sub-graph (a) depicts the training phase of LMAR, which encourages the compensated low-resolution input to maintain representation consistency with the full-resolution UHD input as perceived by the enhancer. The sub-graph (b) demonstrates the inference pipeline of our LMAR, where the compensated low-resolution input is directly fed into the enhancer and then upsampled to the UHD result. The sub-graph (c) illustrates how LMAR works, where the core lies in estimating compensatory kernels under the guidance of model knowledge to make up for the resampling process.

3. Extension on Low-resolution Dataset

In addition to the previously mentioned two UHD image enhancement datasets, we also assessed the effectiveness of our method using the low-resolution LOL [2] dataset. We utilize the synthetic dataset from the LOLv2 version for training and testing, where 900 paired images are used for training and 100 images for testing. Similarly, we first train an enhancer and then correlate the enhancer with the resizer. We retrain the CNN-I on this dataset for 45 epochs with a patch size of 256×256 and batch size of 4. The initial learning rate is $5e^{-4}$, which decays by a factor value of 0.75 every 15 epochs. For the collaborative training phase, we also employ the random scale training strategy on a patch size of 256×256. The training epoch is set to 90 and a batch size of 1. The initial learning rate is $2e^{-4}$, which decays by a factor value of 0.75 every 30 epochs. Since the test image size is 384×384, we make the test on these three downsampling scales, including 192×192, 160×240, and 128×128. We employ two types of resizers to demonstrate the results, including the bicubic and the lanczos3. As shown in Table 1, our method consistently achieved performance improvements across different resampling scales on this low-resolution dataset. It verifies the robustness and scalability of our designs.

Table 1. Quantitative results on the LOL datasets with *CNN-I* as the backbone. The results with LMAR are shown in gray with better results highlighted in **bold**.

Scales	(384, 384)	(192, 192)	(160, 240)	(128, 128)
cubic	21.95 / 0.8907	20.48 / 0.8536	20.42 / 0.8340	19.42 / 0.7552
	21.95 / 0.8906	**20.54 / 0.8583**	**20.50 / 0.8344**	**19.59 / 0.7582**
lanczos3	21.95 / 0.8907	20.40 / 0.8563	20.34 / 0.8313	19.35 / 0.7489
	21.94 / 0.8907	**25.49 / 0.8582**	**20.46 / 0.8335**	**19.53 / 0.7564**

4. Additional Visual Comparison

In this section, we will present more visual comparison results on the UHD-LOL4K dataset and the 4KIL dataset, including the downscaled representations comparison and the final enhanced UHD results comparison.

5. Further Investigations

Our method encounters limitations in handling extremely dark conditions, which could be explored in the future. Additionally, while our method focuses on customizing resampling by spatial domain compensation, the exploration of correlating interpolation resampler and enhancement models in the frequency domain remains unexplored, which presents an intriguing avenue for further investigation.

References

[1] Liangyu Chen, Xin Lu, Jie Zhang, Xiaojie Chu, and Chengpeng Chen. Hinet: Half instance normalization network for image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 182–192, 2021. 1

[2] Wenhan Yang Jiaying Liu Chen Wei, Wenjing Wang. Deep retinex decomposition for low-light enhancement. In *British Machine Vision Conference*, 2018. 2

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[4] Qiaowanni Lin, Zhuoran Zheng, and Xiuyi Jia. Uhd low-light image enhancement via interpretable bilateral learning. *Information Sciences*, 2022. 1

[5] Yang Liu, Zhenyue Qin, Saeed Anwar, Pan Ji, Dongwoo Kim, Sabrina Caldwell, and Tom Gedeon. Invertible denoising network: A light solution for real noise removal. *arXiv preprint arXiv:2104.10546*, 2021. 1

[6] Thomas Müller. tiny-cuda-nn, 2021. 1

[7] Tao Wang, Kaihao Zhang, Tianrun Shen, Wenhan Luo, Bjorn Stenger, and Tong Lu. Ultra-high-definition low-light image enhancement: A benchmark and transformer-based method. *arXiv preprint arXiv:2212.11548*, 2022. 1

[8] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *CVPR*, 2022. 1

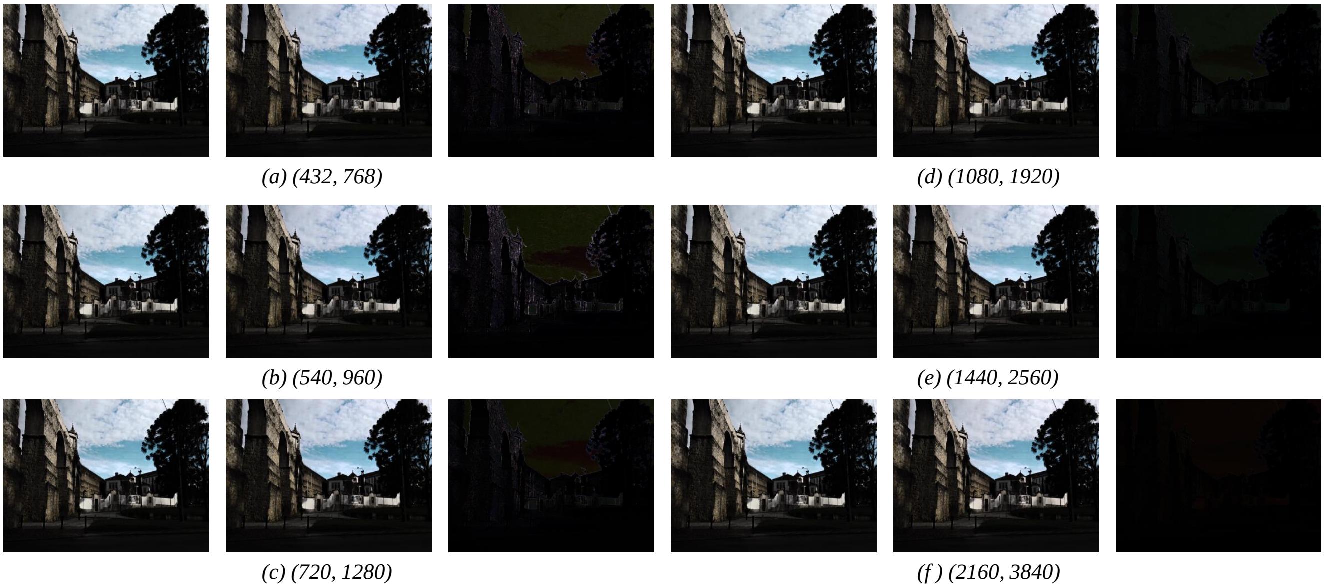

Figure 3. Downscaled representation comparison on the UHD-LOL4K dataset over different scales. The difference is magnified by four times. Please zoom in for details.

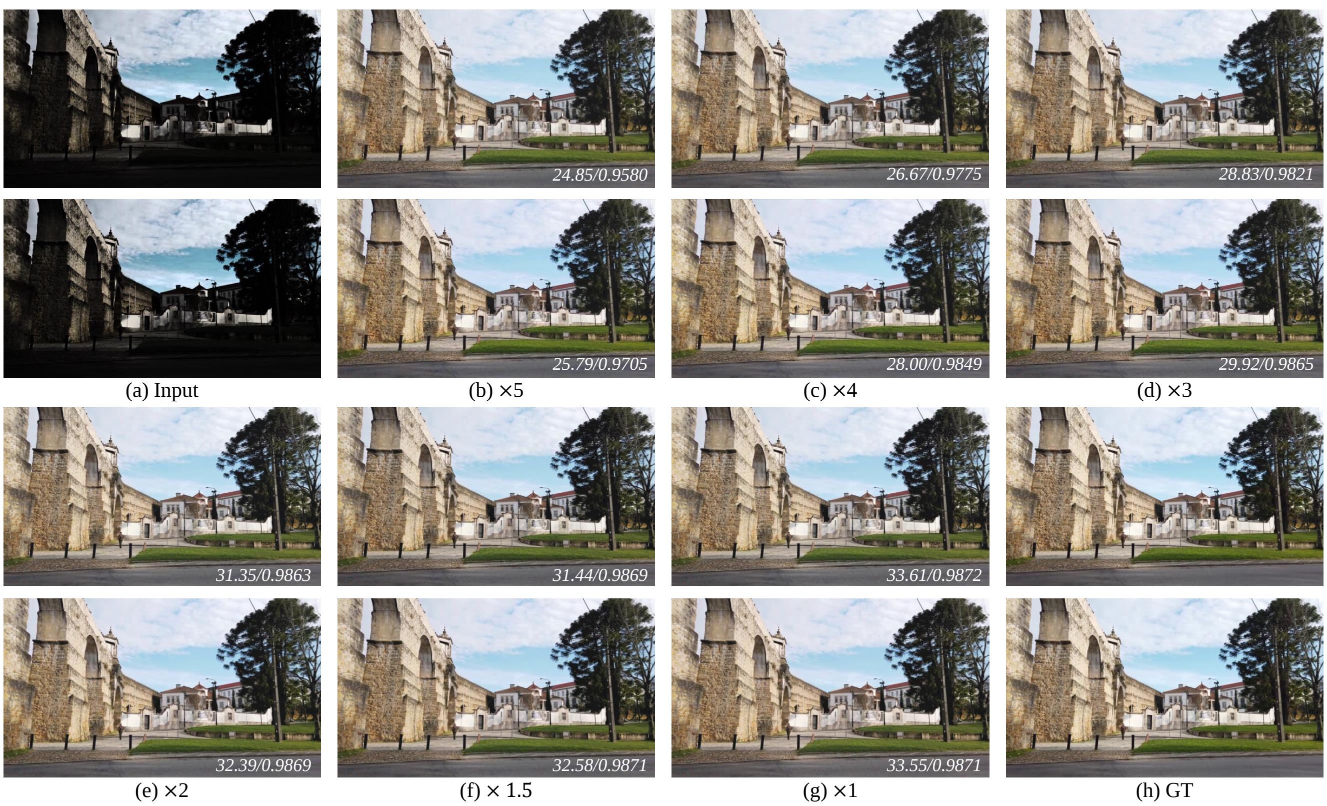

Figure 4. Qualitative results on the UHD-LOL4k dataset of different scales. The top one is obtained from the cubic operator without LMAR, while the bottom one is with LMAR. Please zoom in for details.

Figure 5. Downscaled representation comparison on the UHD-LOL4K dataset over different scales. The difference is magnified by four times. Please zoom in for details.

Figure 6. Qualitative results on the UHD-LOL4k dataset of different scales. The top one is obtained from the cubic operator without LMAR, while the bottom one is with LMAR. Please zoom in for details.

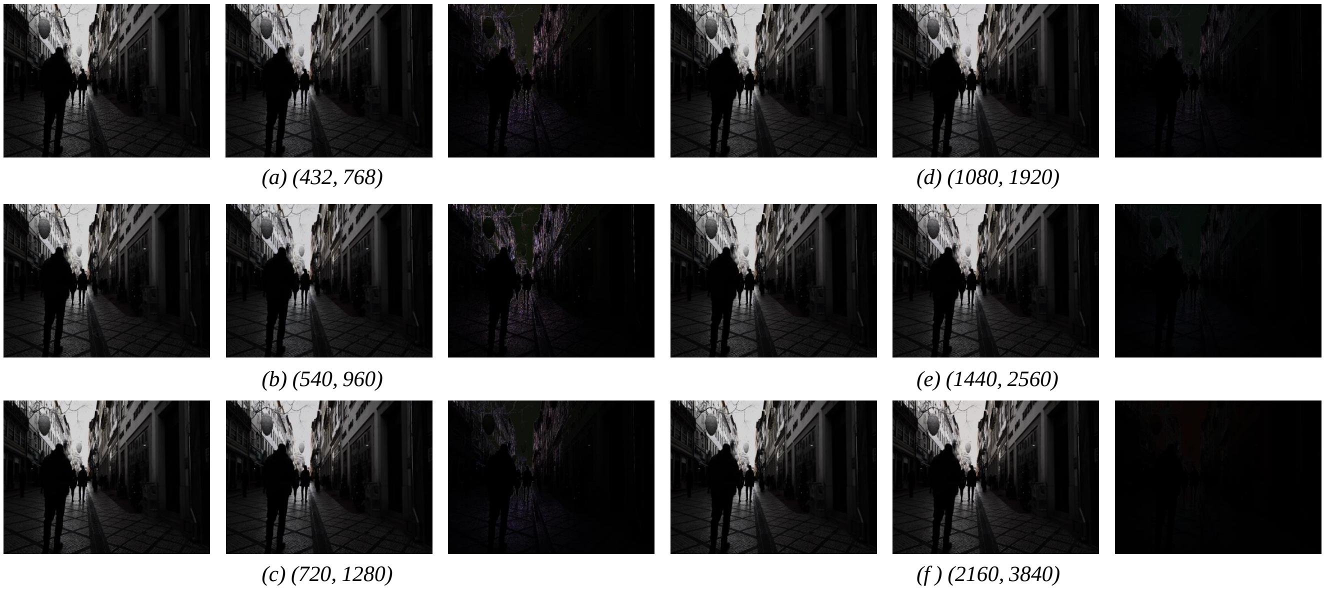

Figure 7. Downscaled representation comparison on the UHD-LOL4K dataset over different scales. The difference is magnified by four times. Please zoom in for details.

Figure 8. Qualitative results on the UHD-LOL4k dataset of different scales. The top one is obtained from the cubic operator without LMAR, while the bottom one is with LMAR. Please zoom in for details.

CVPR
#3204

(a) (432, 768) *(d) (1080, 1920)*

(b) (540, 960) *(e) (1440, 2560)*

(c) (720, 1280) *(f) (2160, 3840)*

Figure 9. Downscaled representation comparison on the 4KIL dataset over different scales. The difference is magnified by four times. Please zoom in for details.

(a) Input (b) ×5 (c) ×4 (d) ×3

(e) ×2 (f) × 1.5 (g) ×1 (h) GT

Figure 10. Qualitative results on the 4KIL dataset of different scales. The top one is obtained from the cubic operator without LMAR, while the bottom one is with LMAR. Please zoom in for details.