# Rethinking Data Augmentation for Image Super-resolution: A Comprehensive Analysis and a New Strategy

Jaejun Yoo*
EPFL.
jaejun.yoo88@gmail.com

Namhyuk Ahn*
Ajou University
aa0dfg@ajou.ac.kr

Kyung-Ah Sohn†
Ajou University
kasohn@ajou.ac.kr

## A. Implementation Details

**Network modification.** To apply CutBlur, the resolution of the input and output has to match. To satisfy such requirement, we first upsample the input $x_{LR} \in \mathbb{R}^{W \times H \times C}$ to $x^s_{LR} \in \mathbb{R}^{sW \times sH \times C}$ using *bicubic* kernel then feed it to the network. In order to achieve efficient inference, we attach *desubpixel* layer [9] at the beginning of the network. By adapting this layer, input is reshaped as $x^s_{LR} \in \mathbb{R}^{W \times H \times s^2 C}$ so that the entire forward pass is performed on the low resolution space. Note that such modifications are only for the synthetic SR task because the other low-level tasks (*e.g.* denoising) have an identical input and output size.

Table 2 shows the performance of original and modified networks. For both RCAN and EDSR, modified networks reach the performance of the original one with negligible increases in the number of the parameters and the inference time. Note that we measure the inference time on the NVIDIA V100 GPU using a resolution of 480×320 for the LR input so that the network generates a 2K SR image.

**Augmentation setup.** Detailed description and setting of every augmentation that we used are described in Table 1. Here, *CutMixup*, *CutBlur*, and *MoA* are the strategies that we have newly proposed in the paper. The hyper-parameters are described following the original papers' notations.

Unless mentioned, at each iteration, we always apply MoA ($p = 1.0$) and evenly choose one method from the augmentation pool. However, we set $p = 0.2$ for training SRCNN and CARN on the synthetic SR dataset and $p = 0.6$ for all the other models for denoising and compression artifact removal tasks, *i.e.*, MoA is applied less. For the realistic SR task (RealSR dataset), we adjust the ratio of MoA to have CutBlur 40% chance more than the other DA's, each of which has 10% chance ($40\% + 10\% + 10\% + 10\% + 10\% + 10\% + 10\% = 100\%$).

**Evaluation protocol.** To evaluate the performance of the model, we use three metrics: peak-signal-to-noise ratio (PSNR), structural similarity index (SSIM) [12], and learned perceptual image patch similarity (LPIPS) [16]. PSNR is defined using the maximum pixel value and mean-squared error between the two images in the log-space. SSIM [12] measures the structural similarity between two images based on the luminance, contrast and structure. Note that we use Y channel only when calculating PSNR and SSIM unless otherwise specified.

Although high PSNR and high SSIM of an image are generally interpreted as a good image restoration quality, it is well known that these metrics cannot represent human visual perception very well [16]. LPIPS [16] has been recently proposed to address this mismatch. It measures the diversity of the generated images using the L1 distance between features extracted from the pre-trained AlexNet [5], which gives better perceptual distance between two images than the traditional metrics. For more details, please refer to the original paper [16].

## B. Detailed Analysis

In this section, we describe each experiment that has been introduced in the Analysis Section. We also provide the results of feature augmentation methods and the original cutout that we excluded in the main text.

**Augmentation in feature space.** We apply feature augmentations [8, 13] to EDSR [6] and RCAN [17] (Figure 1). Both Manifold Mixup [8] and ShakeDrop [13] result in inferior performance than the baselines without any augmentation. For example, RCAN fails to learn with both Manifold Mixup and ShakeDrop. For EDSR, Manifold Mixup is the only one that can be accompanied with, but it also shows significant performance drop. The reason for the catastrophic failure of ShakeDrop is because it manipulates the training signal too much, which induces serious gradient exploding.

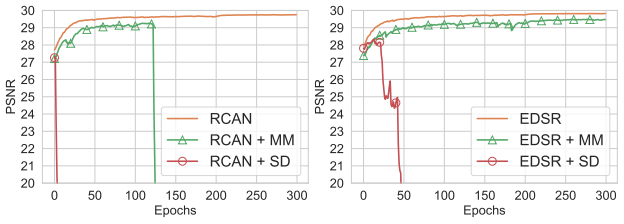
**Cutout.** As discussed in the paper, using the original Cutout [2] setting seriously harms the performance. Here, we demonstrate how Cutout ratio affects the performance (Figure 2). Removing 0.1% of pixels shows similar performance to the baseline, but increasing the dropping proportion to 25% results a huge degradation.

Table 1. A description of data augmentations that are used in our final proposed method.

| Name | Description | Default $\alpha$ |
|---|---|---|
| Cutout [2] | Erase (zero-out) randomly sampled pixels with probability $\alpha$. Cutout-ed pixels are discarded when calculating loss by masking removed pixels. | 0.001 |
| CutMix [14] | Replace randomly selected square-shape region to sub-patch from other image. The coordinates are calculated as: $r_x = \text{Unif}(0, W)$, $r_w = \lambda W$ , where $\lambda \sim \text{N}(\alpha, 0.01)$ (same for $r_y$ and $r_h$). | 0.7 |
| Mixup [15] | Blend randomly selected two images. We use default setting of Feng *et al.* [3] which is: $I' = \lambda I_i + (1-\lambda) I_j$ , where $\lambda \sim \text{Beta}(\alpha, \alpha)$. | 1.2 |
| CutMixup | CutMix with the Mixup-ed image. CutMix and Mixup procedure use hyper-parameter $\alpha_1$ and $\alpha_2$ respectively. | 0.7 / 1.2 ($\alpha_1$ / $\alpha_2$) |
| RGB perm. | Randomly permute RGB channels. | - |
| Blend | Blend image with vector $\mathbf{v} = (v_1, v_2, v_3)$ , where $v_i \sim \text{Unif}(\alpha, 1)$. | 0.6 |
| CutBlur | Perform CutMix with same image but different resolution, producing $\hat{x}_{HR \to LR}$ and $\hat{x}_{LR \to HR}$. Randomly choose $\hat{x}$ from the [$\hat{x}_{HR \to LR}$, $\hat{x}_{LR \to HR}$], then provided selected one as input of the network. | 0.7 |
| MoA (Mixture of Augmentations) | Use all data augmentation method described above. Randomly select single augmentation from the augmentation pool then apply it. | - |

Table 2. Performance (PSNR) and the model size (# parameters and inference time) comparison between the original (*ori.*) and modified (*mod.*) networks on $\times 4$ scale SR dataset. We borrow the reported scores from the performance of the original networks.

| Model | # Params. | Time | Set14 | Urban | Manga |
|---|---|---|---|---|---|
| RCAN (ori.) | 15.6M | 0.612s | 28.87 | 26.82 | 31.22 |
| RCAN (mod.) | 15.6M | 0.614s | 28.86 | 26.76 | 31.24 |
| EDSR (ori.) | 43.1M | 0.334s | 29.80 | 26.64 | 31.02 |
| EDSR (mod.) | 43.2M | 0.335s | 28.81 | 26.66 | 31.06 |




(a) RCAN (b) EDSR

Figure 1. PSNR (dB) comparison on ten DIV2K ($\times 4$) validation images during training. *MM* and *SD* denote the model with Manifold Mixup [8] and ShakeDrop [13], respectively.

## C. Experiment Details

**CutBlur vs. Giving HR inputs during training.** For fair comparison, we provide HR images with $p = 0.33$ otherwise LR images. More specifically, we set the probability of giving HR input, $p$ to 0.33, which is the same ratio to the average proportion of the HR region used in CutBlur.

**Super-resolve the high resolution image.** We quantitatively compare the performance of the baseline and

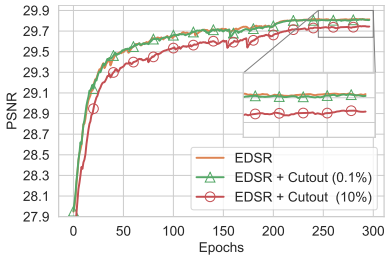



Figure 2. PSNR (dB) comparison between the baseline and two Cutout [2] settings on ten DIV2K ($\times 4$) validation images during training. The gap between two curves varies around 0.1∼0.2 dB.

CutBlur-trained model when the network takes HR images as input in the test phase (Table 4) and when the network takes CutBlurred LR input (Table 5). Here, we generated CutBlurred image by substituting half of the upper part of the LR to its ground truth HR image. When the network takes HR images as input, an ideal method should maintain the input resolution, which would yield infinite (dB) PSNR and 1.0 SSIM. However, the baseline (w/o CutBlur) results in a degraded performance because it tends to over-sharpen the images. This is because the model learns to blindly super-resolve every given pixel. On the other hand, our proposed method provides the near-identical image. when the network takes CutBlurred LR input, the performance of the models without CutBlur are worse than the baseline (bicubic upsample kernel). In contrast, our methods achieve better performance than both the baseline (bicubic) and the models trained without CutBlur.

Such observations are consistently found when using the mixture of augmentations. Note that although the model

Table 3. Quantitative comparison (PSNR / SSIM / LPIPS) on the photo-realistic SR task using generative models. As a baseline, we use ESRGAN [10], which shows state-of-the-art performance on this task.

| Dataset | ESRGAN [10] | ESRGAN [10] + ours |
|---|---|---|
| | PSNR↑ / SSIM↑ / LPIPS↓ | |
| Set14 | 26.11 / 0.6937 / 0.143 | **26.35 / 0.7016 / 0.135** |
| B100 | 25.39 / 0.6522 / 0.177 | **25.49 / 0.6556 / 0.172** |
| Urban100 | 24.49 / 0.7364 / 0.129 | **24.54 / 0.7383 / 0.127** |
| DIV2K | 26.52 / 0.7421 / 0.117 | **26.68 / 0.7448 / 0.114** |

without CutBlur (use all the augmentations except CutBlur) can improve the generalization ability compared to the vanilla EDSR, it still fails to learn such good properties of CutBlur. Only when we include CutBlur as one of the augmentation pool, the model could learn not only "how" but also "where" to super-resolve an image while boosting its generalization ability in a huge margin.

**GAN-based SR models.** We also apply MoA to the GAN-based SR network, ESRGAN [11] and investigated the effect. ESRGAN is designed to produce photo-realistic SR image by adopting adversarial loss [4]. As shown in Table 3, ESRGAN with proposed method outperforms the baseline for both distortion- (PSNR and SSIM) and perceptual-based (LPIPS) metrics. Such result implies that our method adequately enhance the GAN-based SR model as well, considering the perception-distortion trade-off [1].

**Gaussian denoising (color).** To simulate the over-smoothing problem in the denoising task, we conduct a cross-level benchmark test (Table 7) on various noise levels ($\sigma = [30, 50, 70]$) using EDSR [6] and RDN [18] models. In this setting, we test the trained networks on an unseen noise-level dataset. We would like to emphasize that such scenario is common since we cannot guarantee that distortion information are provided in advance in real-world applications. Here, we apply Gaussian noise to the color (RGB) image when we generate a dataset, and PSNR and SSIM are calculated on the full-RGB dimension.

When we train the model on a mild noise level and test to a severe noise (*e.g.* $\sigma = 30 \rightarrow 50$), both the baseline and proposed models show degraded performance since they cannot fully eliminate a noise. On the other hand, for severe $\rightarrow$ mild scenario, models trained with MoA surpass the baseline on SSIM and LPIPS metrics. Note that the high PSNR scores of the baselines without MoA is due to the over-smoothing, which is preferred by PSNR. This can be easily seen in the additional qualitative results Figure 3. Interestingly, the baseline model tends to generate severe artifacts (4th row, 3rd column) since it handles unseen noise improperly. In contrast, our proposed method does not have such artifacts while effectively recovering clean images.

**JPEG artifact removal (color).** Similar to the Gaussian denoising, we train and test the model with various compression factors ($q = [30, 20, 10]$). To generate a dataset, we compress color (RGB) images with different quality levels. However, unlike the color image denoising task, we use Y channel only when calculating PSNR and SSIM. Quantitative and qualitative results on this task are shown in Table 8 and Figure 4, respectively.

**Super-resolution on unseen scale factor.** We also investigate the generalization ability of our model to the SR task. To do that, we test the models on unseen scale factors ($\times 2$ and $\times 3$). Here, the models are only trained on the $\times 4$ scale (Table 6). Our proposed method outperforms the baseline in various scales and datasets. This tendency is more significant when the train-test mismatch becomes bigger (*e.g.*, scale $\times 2$). Figure 5 shows the qualitative comparison of the baseline and ours. While the baseline model over-sharpens the edges producing embossing artifacts, our proposed method effectively super-resolve LR images of the unseen scale factor during training.

**CutBlur in the wild.** We provide more results on real-world out-of-focus photographs that are collected from web (Figure 6).

Table 4. Quantitative comparison (PSNR / SSIM) on artificial SR setup which gives HR image instead of LR. *Baseline* indicates the quantitative metrics between the input (HR) and ground-truth (HR) images.

| Dataset | Baseline | EDSR | | EDSR + mixture of augmentation | |
|---|---|---|---|---|---|
| | | w/o CutBlur | w/ CutBlur | w/o CutBlur | w/ CutBlur |
| DIV2K | inf. / 1.0000 | 22.61 / 0.7072 | **inf. / 1.0000** | 27.33 / 0.8571 | **65.04 / 0.9999** |
| RealSR | inf. / 1.0000 | 23.23 / 0.7543 | **54.64 / 0.9985** | 24.87 / 0.8028 | **46.83 / 0.9951** |

Table 5. Quantitative comparison (PSNR / SSIM) on artificial SR setup which gives CutBlurred image instead of LR. We generate CutBlurred image by replacing half of the upper region of the LR to HR. *Baseline* indicates the quantitative metrics between the input (CutBlurred) and ground-truth (HR) images.

| Dataset | Baseline | EDSR | | EDSR + mixture of augmentation | |
|---|---|---|---|---|---|
| | | w/o CutBlur | w/ CutBlur | w/o CutBlur | w/ CutBlur |
| DIV2K | 29.91 / 0.8799 | 24.08 / 0.7509 | **00.00 / 0.0000** | 27.90 / 0.8468 | **34.59 / 0.9372** |
| RealSR | 30.60 / 0.8883 | 26.26 / 0.7974 | **32.50 / 0.9143** | 27.31 / 0.8244 | **32.46 / 0.9131** |

Table 6. Performance comparison on the SR task evaluated on the DIV2K and RealSR dataset. We train the model using scale factor 4 case and test to scale factor 2 and 3.

| Model | Test Scale | Train Scale ($\times 4$) | |
|---|---|---|---|
| | | DIV2K | RealSR |
| EDSR | $\times 2$ | 23.75 (+0.00) / 0.7414 (+0.0000) | 27.51 (+0.00) / 0.8273 (+0.0000) |
| + proposed | | 31.27 **(+7.52)** / 0.8970 **(+0.1556)** | 31.61 **(+4.10)** / 0.8985 **(+0.0712)** |
| EDSR | $\times 3$ | 27.62 (+0.00) / 0.8142 (+0.0000) | 29.44 (+0.00) / 0.8467 (+0.0000) |
| + proposed | | 28.40 **(+0.78)** / 0.8170 **(+0.0028)** | 29.94 **(+0.50)** / 0.8542 **(+0.0075)** |

Table 7. Performance comparison on the color Gaussian denoising task evaluated on the Kodak24 dataset. We train and test the model on the various noise levels. LPIPS [16] (lower is better) indicates the perceptual distance between the network output and the ground-truth.

| Model | Train $\sigma$ | Test ($\sigma = 30$) | Test ($\sigma = 50$) | Test ($\sigma = 70$) |
|---|---|---|---|---|
| | | PSNR↑ / SSIM↑ / LPIPS↓ | PSNR↑ / SSIM↑ / LPIPS↓ | PSNR↑ / SSIM↑ / LPIPS↓ |
| EDSR | 30 | 31.92 / 0.8716 / 0.136 | 20.78 / 0.3425 / 0.690 | 16.38 / 0.1867 / 1.004 |
| + proposed | | **+0.02 / +0.0006 / -0.004** | **+1.05 / +0.0446 / -0.105** | **+0.35 / +0.0145 / -0.062** |
| RDN | | 31.92 / 0.8715 / 0.137 | 21.61 / 0.3733 / 0.639 | 16.99 / 0.2040 / 0.974 |
| + proposed | | +0.00 / **+0.0002** / +0.000 | -1.01 / -0.0368 / +0.016 | -0.61 / -0.0182 / **-0.020** |
| EDSR | 50 | 29.64 / 0.7861 / 0.306 | 29.66 / 0.8136 / 0.209 | 21.50 / 0.3553 / 0.687 |
| + proposed | | -0.54 / **+0.0708 / -0.158** | +0.00 / -0.0002 / **-0.001** | **+0.26 / +0.0212 / -0.029** |
| RDN | | 29.77 / 0.7931 / 0.298 | 29.63 / 0.8134 / 0.208 | 23.68 / 0.4549 / 0.519 |
| + proposed | | -1.00 / **+0.0544 / -0.146** | -0.01 / -0.0005 / +0.002 | -1.40 / -0.0666 / +0.104 |
| EDSR | 70 | 27.38 / 0.7295 / 0.375 | 27.95 / 0.7385 / 0.366 | 28.23 / 0.7689 / 0.273 |
| + proposed | | -2.51 / **+0.0696 / -0.193** | **+0.46 / +0.0674 / -0.139** | +0.00 / -0.0003 / **-0.002** |
| RDN | | 28.23 / 0.7546 / 0.344 | 28.13 / 0.7517 / 0.349 | 28.19 / 0.7684 / 0.275 |
| + proposed | | -3.48 / **+0.0461 / -0.163** | -0.93 / **+0.0337 / -0.137** | **+0.01** / -0.0003 / **-0.006** |

Table 8. Performance comparison on the color JPEG artifact removal task evaluated on the LIVE1 [7] dataset. We train and test the model on the various quality factors. LPIPS [16] (lower is better) indicates the perceptual distance between the network output and the ground-truth. Unlike the color image denoising task, we use Y channel only when calculating PSNR and SSIM.

| Model | Train $q$ | Test ($q = 30$) | Test ($q = 20$) | Test ($q = 10$) |
|---|---|---|---|---|
| | | PSNR↑ / SSIM↑ / LPIPS↓ | PSNR↑ / SSIM↑ / LPIPS↓ | PSNR↑ / SSIM↑ / LPIPS↓ |
| EDSR | 30 | 33.95 / 0.9227 / 0.118 | 32.36 / 0.8974 / 0.155 | 29.17 / 0.8196 / 0.286 |
| + proposed | | -0.01 / -0.0002 / +0.001 | **+0.02** / **+0.0003** / +0.001 | +0.00 / -0.0005 / +0.001 |
| RDN | | 33.90 / 0.9220 / 0.121 | 32.34 / 0.8971 / 0.157 | 29.20 / 0.8202 / 0.287 |
| + proposed | | **+0.01** / **+0.0003** / **-0.003** | -0.01 / -0.0001 / **-0.001** | -0.05 / -0.0017 / +0.002 |
| EDSR | 20 | 33.64 / 0.9174 / 0.130 | 32.52 / 0.8979 / 0.160 | 29.67 / 0.8327 / 0.271 |
| + proposed | | **+0.18** / **+0.0037** / **-0.008** | +0.00 / **+0.0001** / **-0.001** | +0.00 / -0.0005 / **-0.001** |
| RDN | | 33.59 / 0.9164 / 0.132 | 32.47 / 0.8972 / 0.162 | 29.65 / 0.8322 / 0.271 |
| + proposed | | **+0.14** / **+0.0031** / **-0.005** | +0.00 / **+0.0001** / +0.001 | **+0.01** / -0.0003 / +0.001 |
| EDSR | 10 | 32.45 / 0.8992 / 0.154 | 31.83 / 0.8840 / 0.179 | 30.14 / 0.8391 / 0.254 |
| + proposed | | **+0.97** / **+0.0179** / **-0.020** | **+0.45** / **+0.0104** / **-0.011** | +0.00 / -0.0001 / +0.001 |
| RDN | | 32.37 / 0.8967 / 0.166 | 31.78 / 0.8821 / 0.189 | 30.10 / 0.8381 / 0.259 |
| + proposed | | **+0.95** / **+0.0187** / **-0.023** | **+0.40** / **+0.0106** / **-0.013** | -0.01 / -0.0002 / +0.003 |

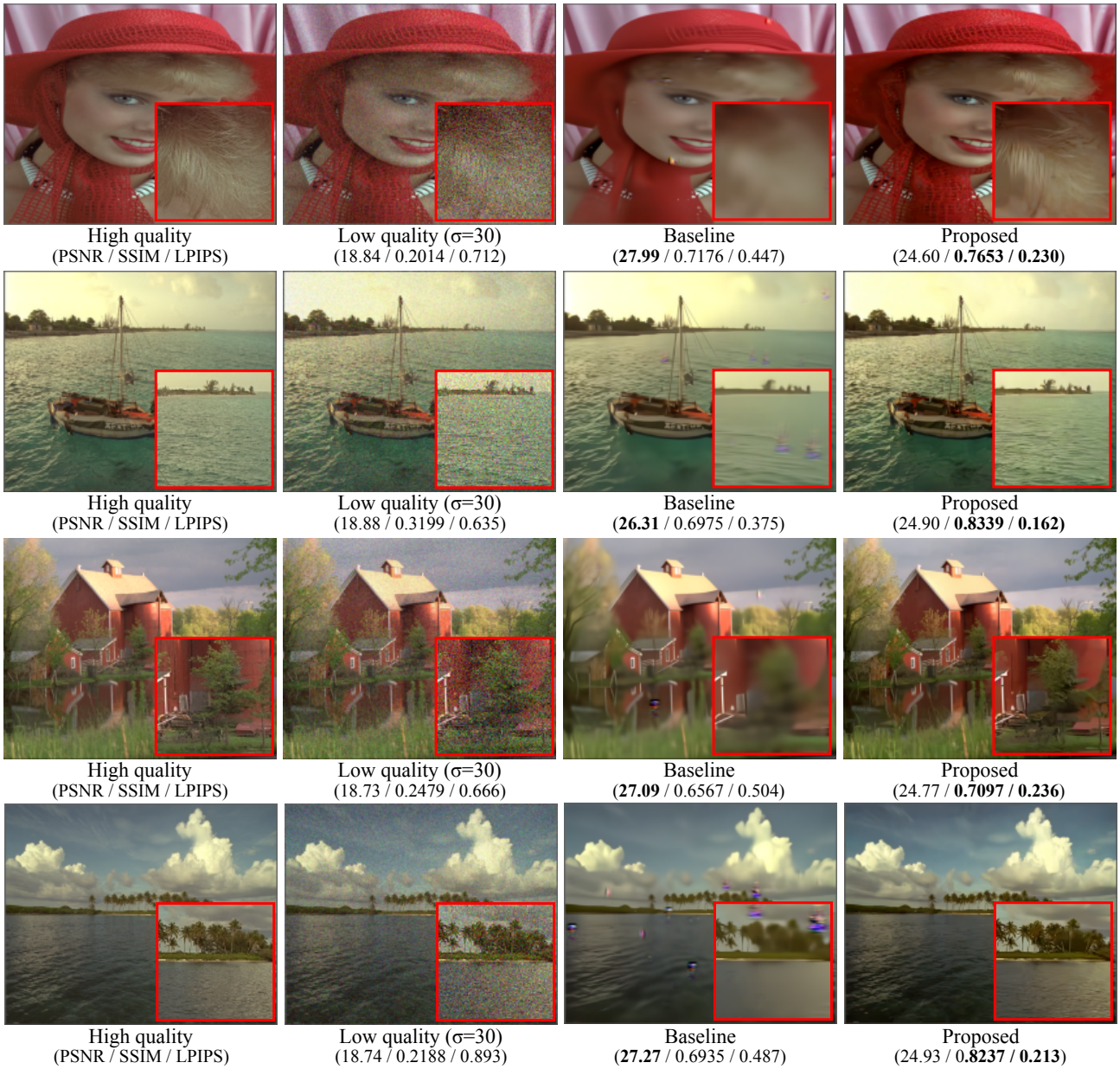



Figure 3. Comparison of the generalization ability on the color Gaussian denoising task. Both methods are trained on severely distorted dataset ($\sigma = 70$) and tested on the mild case ($\sigma = 30$). The baseline over-smooths the inputs or generates artifacts while ours successfully reconstructs the fine structures.

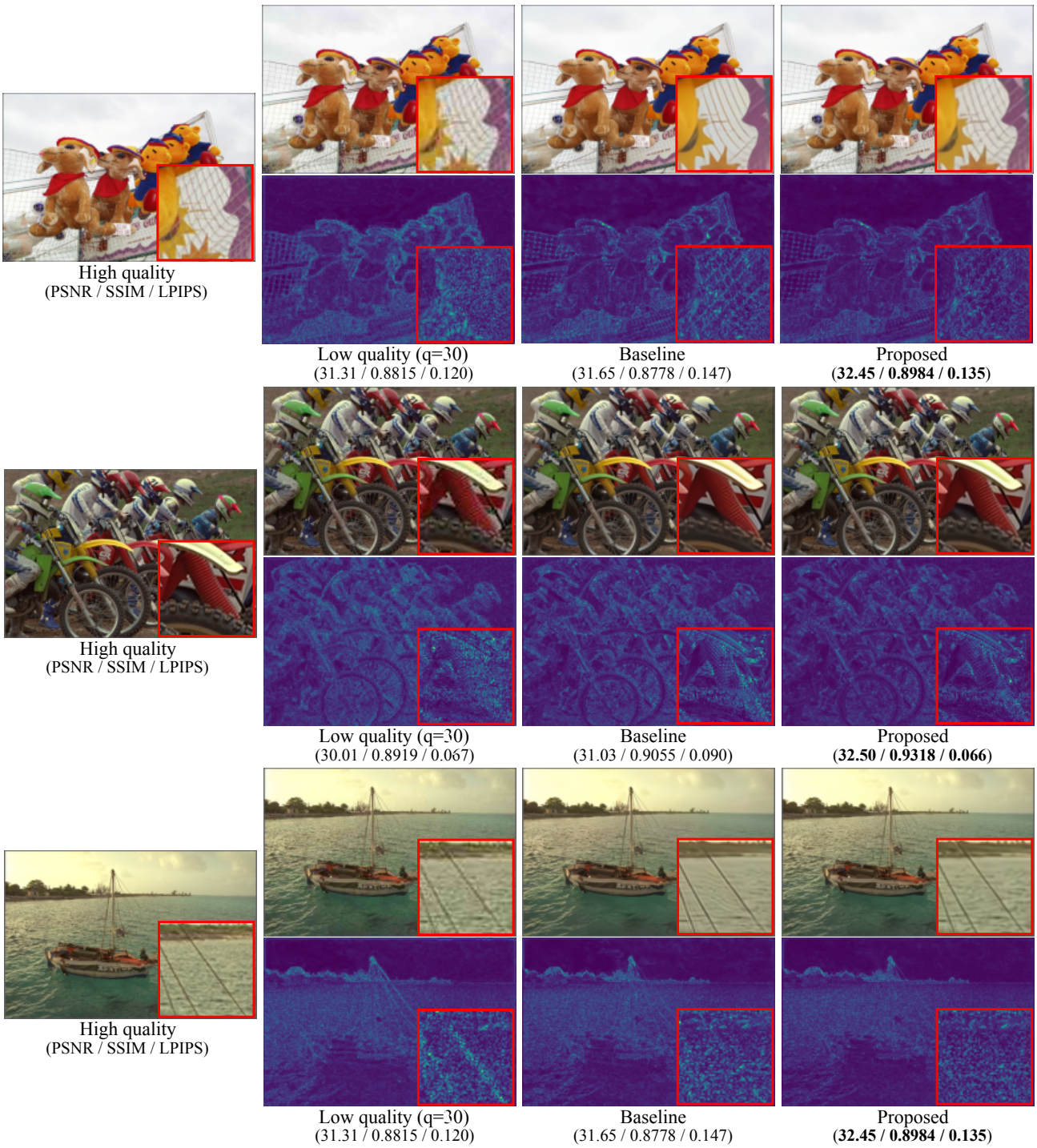



Figure 4. Comparison of the generalization ability on the color JPEG artifact removal task. Both methods are trained on severely compressed dataset ($q = 10$) and tested on the mild case ($q = 30$).

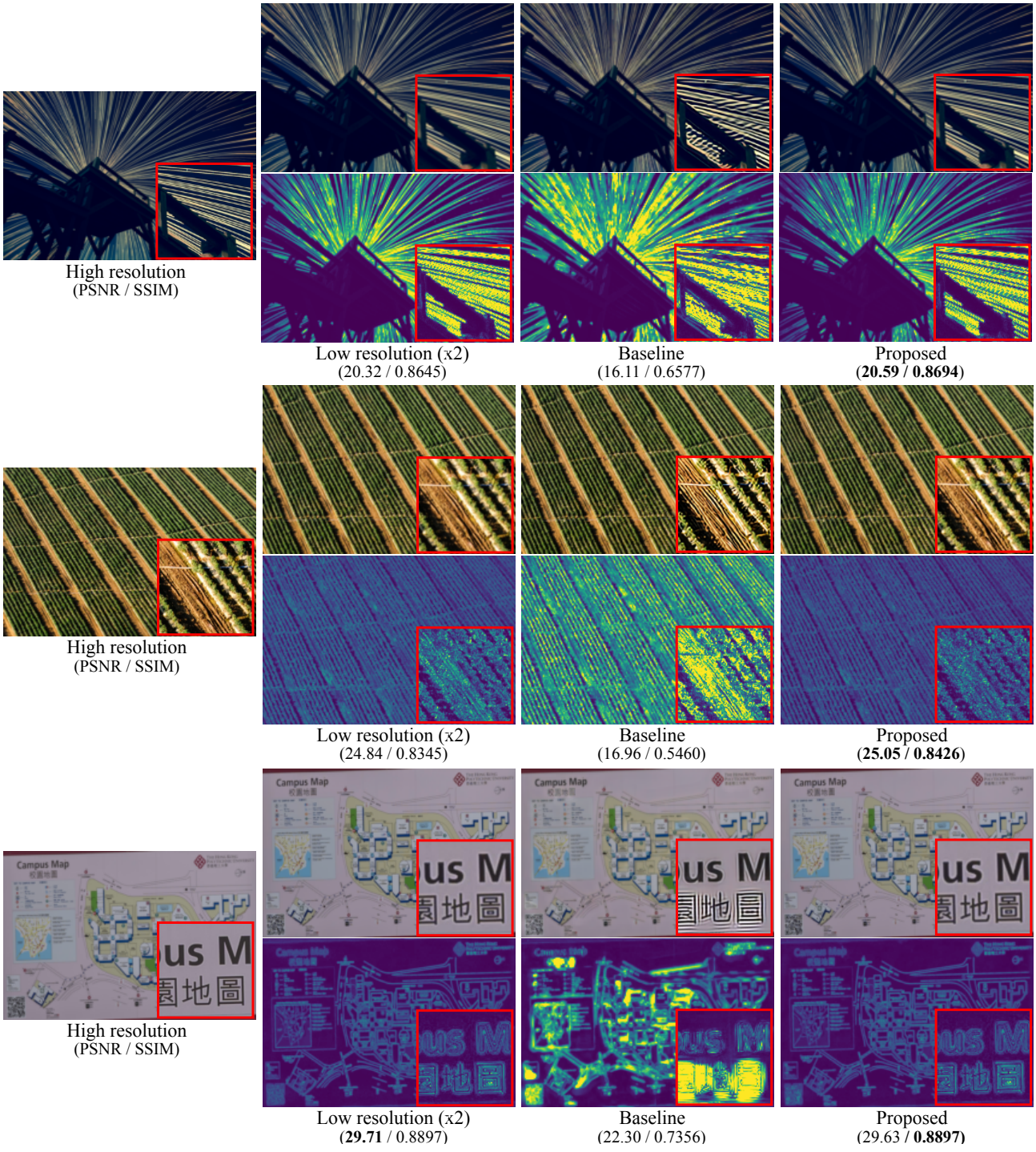



Figure 5. Comparison of the generalization ability on the SR task. Both methods are trained on $\times 4$ scale factor dataset and tested on different scale factor ($\times 2$). The baseline tend to produce the distortion due to the over-sharpening while proposed method does not. Similar to the denoising task, the baseline over-smooths inputs so that it fails to recover fine details.

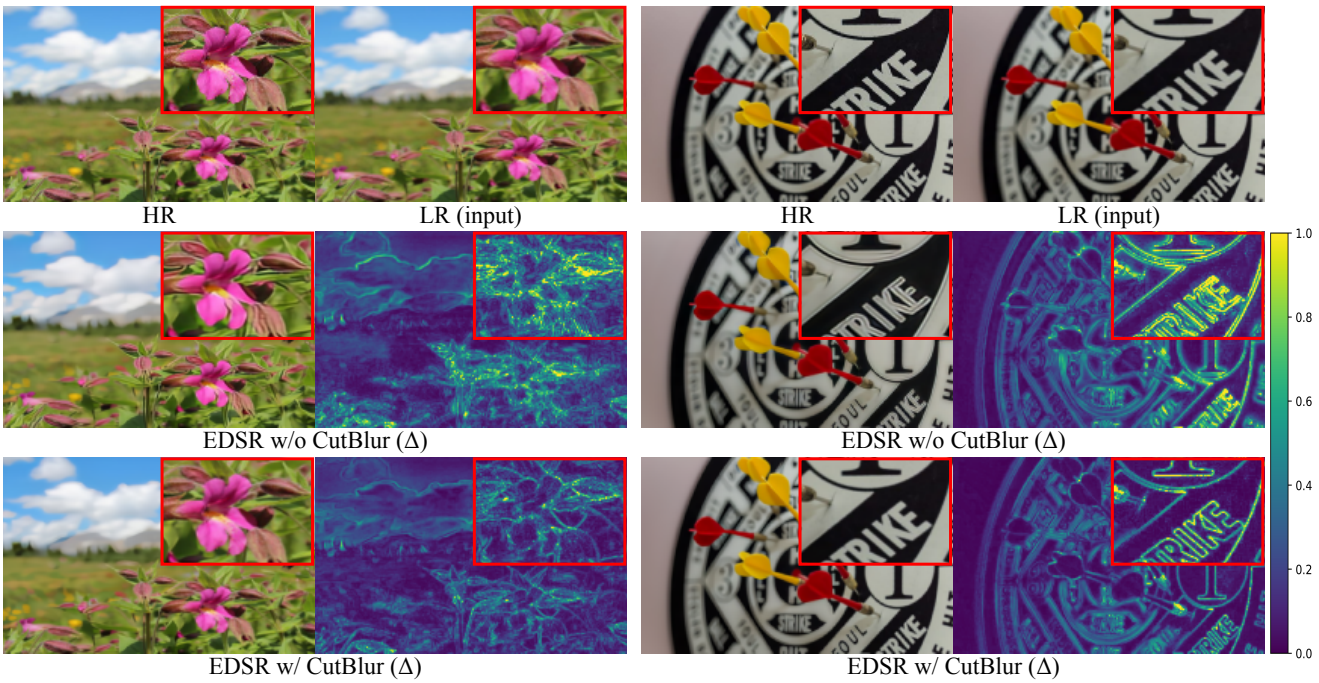



Figure 6. Qualitative comparison of the baseline and CutBlur model outputs. The inputs are the real-world out-of-focus photography ($\times 2$ bicubic downsampled) taken from a web. The baseline model over-sharpens the focused region (foreground) resulting in unpleasant artifact while our method effectively super-resolves the image without generating such distortions.

## References


[1] Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6228–6237, 2018. 3

[2] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 1, 2

[3] Ruicheng Feng, Jinjin Gu, Yu Qiao, and Chao Dong. Suppressing model overfitting for image super-resolution networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 2

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 3

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[6] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017. 1, 3

[7] HR Sheikh. Live image quality assessment database release 2. *http://live. ece. utexas. edu/research/quality*, 2005. 5

[8] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, Aaron Courville, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. *arXiv preprint arXiv:1806.05236*, 2018. 1, 2

[9] Thang Vu, Cao Van Nguyen, Trung X Pham, Tung M Luu, and Chang D Yoo. Fast and efficient image quality enhancement via desubpixel convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 1

[10] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 3

[11] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 3

[12] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 1

[13] Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise. Shakedrop regularization for deep residual learning. *arXiv preprint arXiv:1802.02375*, 2018. 1, 2

[14] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *arXiv preprint arXiv:1905.04899*, 2019. 2

[15] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2

[16] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 1, 4, 5

[17] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 286–301, 2018. 1

[18] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image restoration. *arXiv preprint arXiv:1812.10477*, 2018. 3