Adaptive Dropout: Unleashing Dropout across Layers for Generalizable Image Super-Resolution

Supplementary Material

1. More Implementation Details

1.1. Dataset details

We train our blind SR model on the DIV2K dataset, cropping the high-resolution images into smaller 800×800 images to reduce the network's computational burden. To degrade the high-resolution images into corresponding low-resolution images, we use a widely employed multi-degradation setting. Our training degradation setup remains consistent with baseline methods, using a "second-order" degradation generation setting. In each order, the image has a certain probability of sequentially applying noise, blur, and JPEG degradation types during downsampling to disrupt the original image information maximally.

To test the generalization ability of blind SR models trained on the multi-degradation setting, we adopt the five widely used datasets Set5, Set14, BSD100, Manga109, and Urban100. Different from training degradations, we simply degrade the HR images from these datasets with eight types of synthetic degradations or degradation combinations, including bicubic (abbreviated as clean), bicubic + blur (abbreviated as blur), bicubic + noise (abbreviated as noise), bicubic + jpeg (abbreviated as jpeg), bicubic + blur + noise (abbreviated as b+n), bicubic + blur + jpeg (abbreviated as b+j), bicubic + noise + jpeg (abbreviated as n+j), bicubic + blur + noise + jpeg (abbreviated as b+n+j), where bicubic is bicubic downsampling, b is *blur*, n is *noise*, j is *jpeg*. These degradations include those similar to the training degradations (such as b+n+j) and those far from the training degradations (such as clean, noise). Therefore, these eight degradation types can effectively illustrate the fitting and generalization capabilities of a blind SR model. However, it is important to note that the above method of evaluating the capabilities of blind SR models is only valid on synthetic datasets. Considering that the ultimate goal of blind SR models is to effectively enhance the quality and resolution of photographs in the real world, we also employ several real-world datasets. RealSR and DRealSR are two recent real-world super-resolution datasets, where the low-resolution and high-resolution images are captured by cameras at different focal lengths, which better reflect the capabilities of blind SR models in the real world compared to synthetic datasets. Additionally, to remain consistent with baseline methods, we also use the realistic NTIRE 2018 SR challenge data to demonstrate the generalization capabilities of our method.

1.2. Training details

During the training process, we employ the L1 loss function and the Adam optimizer. The values of β_1 and β_2 of the Adam optimizer are set to 0.9 and 0.999 respectively. The batch size is set to 16, and the low-resolution (LR) images have dimensions of 32×32 pixels. We implement a cosine annealing learning strategy to adjust the learning rate. Initially, the learning rate is set to 2×10^{-4}. The cosine annealing period for adjusting the learning rate spans 500,000 iterations. We train and test all our models using the PyTorch framework and conducted the training on 3090 GPUs.

1.3. Metrics

We primarily use PSNR(Peak Signal-to-Noise Ratio) to measure the performance of the model on different datasets. For real-world datasets, we also use LPIPS(Learned Perceptual Image Patch Similarity) to evaluate the perceptual quality of the restored high-resolution images, which is more important for real-world images. We present the relevant LPIPS results in the supplementary materials.

2. Detailed Comparisons

We show detailed comparison data with Simple-Align in Table 4. We outperform Simple-Align on almost all degradations. The result is understandable for we consider the importance of explicit generalization for intermediate layers. We show more visual results of SRResNet with different regularization methods in Figure 4, Figure 5, Figure 6, and Figure 7.

Table 1. **The performance of SRResNet on Set5 and Set14 with different annealing strategies.** These annealing strategies are all for Explicit Adaptive Dropout and are explicitly adopted.

dropout format	training strategy	psnr	
		Set5	Set14
standard dropout	None	24.33	22.45
	linear annealing	24.55	22.83
	layer-wise annealing	**24.89**	**23.01**
	linear+layer-wise	24.81	22.94
adaptive dropout	None	25.89	23.36
	linear annealing	25.85	23.26
	layer-wise annealing	**26.07**	**23.46**
	linear+layer-wise	25.91	23.32

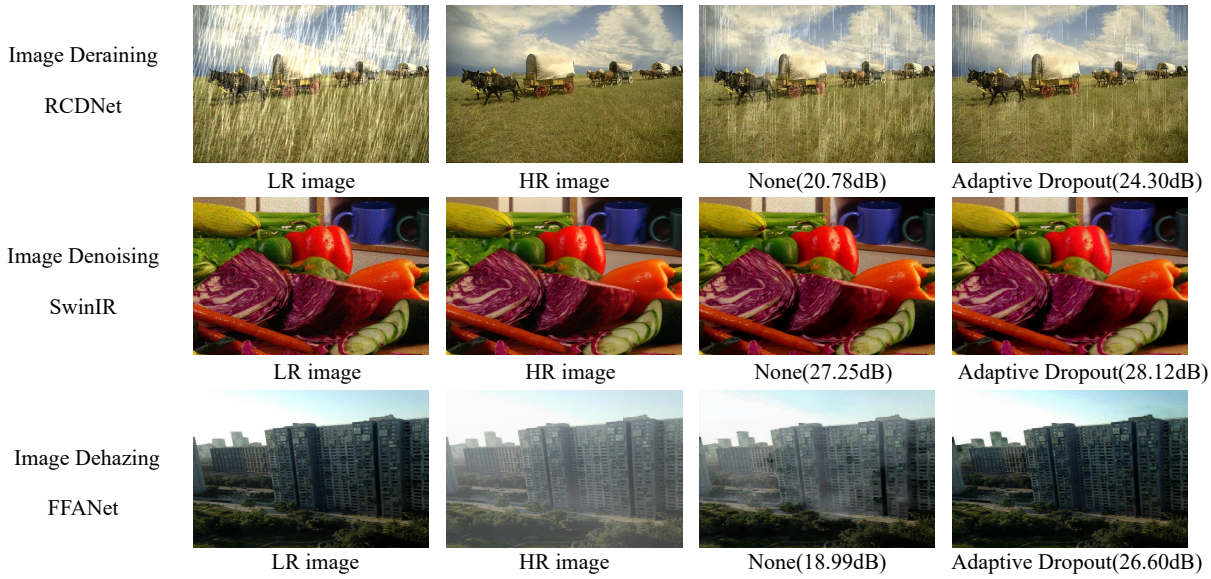

Figure 1. **Visual comparison of baseline w or w/o our regularization methods in single-degradation tasks.**

Table 2. **The performance of SRResNet with different formats of w.** *nth block* means that before nth block, we use $value\ w$ and after nth block, we use $vector\ w$.

nth block	model performance					
	Set5	Set14	BSD100	Manga109	Urban100	average
0	24.61	22.55	23.00	19.13	20.67	21.99
4	24.58	22.56	22.94	19.23	20.69	22.00
8	24.55	22.50	22.94	19.29	20.81	**22.09**
12	24.37	22.40	22.75	19.40	20.78	21.94
16	24.28	22.36	22.86	19.44	21.02	21.98

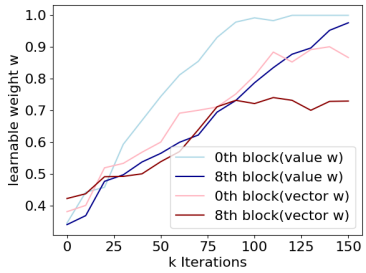

Figure 2. **The trend of the average value of vector w and value w over the training process.** Vector w is more difficult to converge to 1 compared to value w.

3. More discussions

3.1. About the annealing strategy for Explicit Adaptive Dropout

For Explicit Adaptive Dropout, we use a layer-wise annealing strategy as the adaptive training strategy to help blind SR models generalize better. A competing annealing strategy involves reducing the weighted dropout w by the same amount in every layer at regular intervals until w reaches 0. However, we found that this strategy does not account for the different levels of sensitivity to perturbations between shallow and deep layers of the network, resulting in poor generalization. Additionally, we also attempt to gradually reduce the w corresponding to the block being annealed while performing layer-wise annealing. Specifically, originally during layer-wise annealing, we keep w constant until t iterations, after which we set the w for that block to 1. Now, we linearly decrease w to 0 over these t iterations. We find that although this fully follows the pattern we discover in the learnable weighted structure, it does not improve the model's generalization capability and performs slightly worse than simple layer-wise annealing. This indicates that when adding regularization to the intermediate layers of the network, it is more important to consider the different requirements of each layer for regularization. For the learnable weighted structure, gradually reducing w to 1 primarily satisfies the need for the network to fit the training set, which may not necessarily have a positive impact on generalization performance. Therefore, when testing our two designed variants on the blind SR model, we often find that the explicit weighted dropout outperforms the implicit weighted dropout. This suggests that gradually reducing w to 1 is not essential; what is more important is the relative relationship of w between different layers. We show the

performance of SRResNet on Set5 and Set14 with different training strategies in Table 1.

Table 3. **The average performance of SRResNet on the five synthetic datasets.** We only calculate the average on the first four degradations(clean, noise, blur, jpeg), considering they best reflect the network's generalization capability. *plan A* is using *value w* for shallow layers and using *vector w* for deep layers. *plan B* is the opposite one.

postion	average performance
plan A	**22.09**
plan B	22.01

3.2. About vector w and value w

For Implicit Adaptive Dropout, we use different formats of w for different layers. Specifically, we use `value w` for shallow layers and `vector w` for deep layers. Here, we explain the specific reasons. For `vector w`, we plot the average value of w corresponding to that layer over the training process as a curve and display it alongside the w for `value w` in Figure 2. Compared to `value w`, the average value of `vector w` is harder to converge. This indicates that this perturbation is stronger and affects the generalization of specific degradations. The perturbation given by `value w` is weaker and affects more general generalization. If the perturbation in shallow layers is too strong, it can affect the stability of network training. Therefore, we use `value w` in shallow layers and `vector w` in deep layers, achieving a good trade-off between fitting and generalization.

Furthermore, we conduct ablation experiments to determine from which layer to switch from `value w` to `vector w` and show the result in Table 2. It can be observed that using only `vector w` or `value w` alone cannot perform well across all datasets. Combining both achieves a better balance in different scenarios. Specifically, using `vector w` allows for the allocation of appropriate w to different features, better addressing the imbalance between channels, thereby forming a more robust general representation that is not only applicable to different degradations but also to different scenes. Using `value w` reduces the network parameters for learning w, with all channels sharing one w. While this is not conducive to alleviating the imbalance between channels, it ensures the network's fitting ability, which to some extent is a result of this imbalance. Manga100 and Urban100 datasets in the test set have distributions significantly different from the training set, so `vector w` performs better on these datasets. Conversely, the Set5, Set14, and BSD100 datasets are relatively close to the training set, and using `value w` may to some extent help restore image quality in these datasets. Therefore, we combine both, using them at different layers, to achieve good generalization performance across all datasets. We decide to switch from `value w` to `vector w` at the mid-

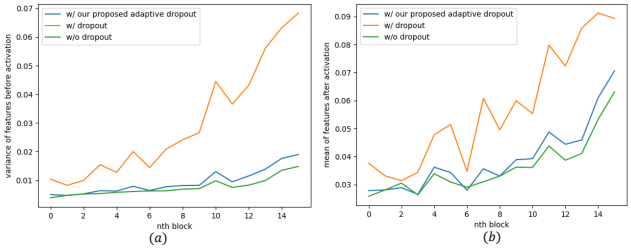

Figure 3. **statistics of features with or without dropout during training.** (a) shows that dropout alters the variance of features although it keeps the mean. (b) shows that the mean of features also changes after activation when they undergo dropout before.

point of the network, which exhibits the best performance as shown in Table 2. Additionally, we conduct ablation experiments on whether to use `value w` or `vector w` in shallow or deep layers in Table 3, which demonstrate that using `value w` in shallow layers and `vector w` in deep layers can lead to better generalization performance.

3.3. About the inconsistency during training

As illustrated in section 3.1 of Li et al. [5] or section 2.1 of Kim et al. [3], the variance of features has been altered after dropout during training. And then after non-linear activation functions, the variance shift leads to the mean shift. We also provide the statistical changes in features during training in Fig. 3, which can be alleviated with our proposed adaptive dropout.

3.4. About the relationship between the two variants

For Adaptive Dropout, we design two variants based on how they adopt the adaptive training strategy. Through experimental results, it can be observed that the two variants exhibit different performances across various degradations and datasets. The biggest difference between the two variants lies in the different ways they utilize the adaptive training strategy. On the basis of layer-wise annealing, Explicit adaptive dropout explicitly employs this strategy, maintaining w constant before annealing to provide greater perturbation to intermediate layer features. In contrast, Implicit Adaptive Dropout implicitly utilizes this strategy while striving to ensure the network's fitting ability.

Additionally, there is also room for combining these two variants. In Implicit Adaptive Dropout, we also observe similar annealing phenomena as in Figure 2. However, directly combining the two variants is somewhat challenging, so we will explore this in future work, where explicit annealing in shallow layers and implicit annealing in deep layers may be an area worth exploring.

3.5. About extensions on single-degradation tasks

We tested our regularization method on three tasks: image deraining, image denoising, and image dehazing. We use

Table 4. **The PSNR (dB) results of models with** $\times 4$**.** We test them on eight types of degradations or multi-degradations(b is $blur$, n is $noise$, j is $jpeg$). **Red** and Blue indicate the best and the second-best performance, respectively.

Models	Regularization	Set5 [1]		Set14 [10]		BSD100 [7]		Manga109 [8]		Urban100 [2]	
		clean	blur	clean	blur	clean	blur	clean	blur	clean	blur
SRResNet [4]	None	24.89	24.76	22.60	22.50	23.06	22.99	18.42	18.75	21.24	21.06
	Simple-Align [9]	25.94	25.45	23.46	23.18	23.69	23.47	19.34	19.50	21.83	21.40
	Explicit Adaptive Dropout	**26.11**	25.39	23.42	23.20	**23.76**	**23.59**	19.40	19.62	21.87	**21.48**
	Implicit Adaptive Dropout	26.09	**25.56**	**23.47**	**23.21**	23.73	23.50	**19.51**	**19.69**	**21.90**	21.40
RRDB [11]	None	25.21	25.14	22.98	22.65	23.38	23.31	18.59	18.64	21.57	21.17
	Simple-Align [9]	26.46	26.27	23.76	23.59	23.90	23.94	19.21	19.45	22.21	**21.94**
	Explicit Adaptive Dropout	**26.64**	26.34	23.72	23.69	**24.04**	**24.02**	19.47	19.71	22.23	21.76
	Implicit Adaptive Dropout	26.57	**26.30**	**23.79**	**23.76**	24.00	23.89	**19.53**	**19.79**	**22.12**	21.73
SwinIR [6]	None	26.25	26.03	23.76	23.47	23.91	23.83	19.10	19.19	22.18	21.90
	Simple-Align [9]	26.40	26.15	23.89	23.50	23.97	**23.95**	19.21	19.34	22.27	22.07
	Explicit Adaptive Dropout	**26.54**	**26.21**	23.92	**23.56**	**24.09**	23.59	19.22	19.39	**22.41**	**22.19**
	Implicit Adaptive Dropout	26.46	26.12	**23.95**	23.41	24.01	23.50	**19.31**	**19.44**	22.32	22.16
		noise	jpeg	noise	jpeg	noise	jpeg	noise	jpeg	noise	jpeg
SRResNet [4]	None	22.02	23.72	20.81	21.84	20.34	22.48	19.74	18.30	19.73	20.60
	Simple-Align [9]	22.32	24.33	21.11	22.45	21.46	22.93	18.64	19.05	19.86	21.10
	Explicit Adaptive Dropout	**22.41**	**24.38**	21.14	22.46	21.48	**23.00**	18.67	19.07	**19.90**	21.11
	Implicit Adaptive Dropout	**22.41**	24.36	**21.15**	**22.51**	**21.53**	**23.00**	**18.76**	**19.20**	**19.90**	**21.15**
RRDB [11]	None	21.79	23.86	20.70	22.07	20.98	22.73	18.29	18.44	19.61	20.92
	Simple-Align [9]	22.71	24.56	21.45	22.60	21.76	23.09	18.78	19.08	**20.00**	**21.33**
	Explicit Adaptive Dropout	**22.80**	24.62	24.53	22.67	**21.84**	**23.20**	18.84	19.22	19.94	21.24
	Implicit Adaptive Dropout	22.81	**24.59**	**21.58**	**22.80**	21.71	23.13	**18.87**	**19.23**	19.87	21.23
SwinIR [6]	None	22.96	24.37	21.56	23.04	22.12	23.04	18.71	18.95	20.56	21.32
	Simple-Align [9]	23.49	24.62	21.64	23.19	22.21	23.15	18.90	19.15	20.69	21.37
	Explicit Adaptive Dropout	**23.63**	**24.68**	21.67	**23.29**	**22.24**	**23.29**	18.92	19.27	**20.85**	**21.52**
	Implicit Adaptive Dropout	23.49	24.49	**21.69**	23.19	22.19	23.28	**19.02**	**19.31**	20.74	21.40
		b+n	b+j	b+n	b+j	b+n	b+j	b+n	b+j	b+n	b+j
SRResNet [4]	None	23.31	23.44	21.81	21.70	22.27	22.34	18.60	18.53	20.46	20.30
	Simple-Align [9]	23.64	**23.85**	22.12	22.24	22.43	22.71	19.22	19.14	20.56	20.61
	Explicit Adaptive Dropout	**23.74**	23.80	**22.15**	**22.29**	**22.48**	**22.81**	19.26	19.23	**20.57**	**20.66**
	Implicit Adaptive Dropout	23.65	23.73	22.12	**22.29**	22.44	22.78	**19.51**	**19.29**	20.50	20.63
RRDB [11]	None	23.52	23.48	22.05	21.76	22.40	22.46	18.82	18.44	20.57	20.39
	Simple-Align [9]	23.80	23.95	22.35	22.29	22.63	22.88	19.33	19.21	**20.68**	20.79
	Explicit Adaptive Dropout	**23.83**	24.03	22.38	22.36	**22.67**	**22.93**	19.45	19.28	20.66	**20.79**
	Implicit Adaptive Dropout	23.76	**24.05**	**22.36**	**22.41**	22.63	22.92	**19.48**	**19.28**	20.62	20.71
SwinIR [6]	None	23.80	23.84	22.20	22.26	22.61	22.82	19.07	19.02	20.89	20.79
	Simple-Align [9]	24.13	24.17	22.55	22.32	22.74	22.97	19.23	19.22	**21.02**	**20.98**
	Explicit Adaptive Dropout	**24.24**	24.27	**22.91**	**22.41**	**22.81**	**23.05**	19.25	19.29	21.00	20.96
	Implicit Adaptive Dropout	24.18	**24.30**	22.78	22.36	22.80	**23.05**	**19.31**	**19.37**	20.92	20.94
		n+j	b+n+j	n+j	b+n+j	n+j	b+n+j	n+j	b+n+j	n+j	b+n+j
SRResNet [4]	None	23.21	22.70	21.59	21.44	22.24	22.05	18.25	18.43	20.42	20.10
	Simple-Align [9]	23.71	23.10	22.08	21.83	22.57	22.29	18.93	18.99	20.78	**20.29**
	Explicit Adaptive Dropout	23.71	**23.10**	22.09	**21.84**	22.59	22.31	18.95	19.04	20.79	**20.29**
	Implicit Adaptive Dropout	**23.74**	**23.10**	**22.13**	**21.84**	**22.60**	**22.32**	**19.07**	**19.10**	**20.81**	20.27
RRDB [11]	None	23.48	22.80	21.88	21.59	22.44	22.16	18.42	18.45	20.74	20.25
	Simple-Align [9]	23.85	23.06	22.22	21.85	22.67	22.36	19.00	19.02	**21.00**	20.40
	Explicit Adaptive Dropout	**23.85**	23.12	22.24	21.87	**22.70**	**22.41**	19.05	19.08	20.94	**20.41**
	Implicit Adaptive Dropout	23.87	**23.10**	**22.28**	**21.90**	22.68	22.40	**19.06**	**19.12**	**20.94**	20.40
SwinIR [6]	None	23.67	22.99	22.11	21.82	22.61	22.34	18.79	18.80	20.98	20.45
	Simple-Align [9]	23.80	23.09	**22.33**	21.90	22.76	22.39	**19.02**	19.03	21.12	20.53
	Explicit Adaptive Dropout	**23.82**	**23.18**	22.30	22.19	**22.78**	**22.50**	18.82	18.91	**21.15**	**20.57**
	Implicit Adaptive Dropout	23.76	23.13	22.23	**22.19**	22.57	22.39	18.87	**19.10**	21.02	20.56

Adaptive Dropout and set w to 0.9 in all blocks to simply verify the effectiveness of our method. The specific models, datasets, and results have been presented in the main text. Here, we display the visual results in Figure 1. It can be observed that the baseline fails to restore image quality well when encountering degradations and scenes inconsistent with training. In contrast, Adaptive Dropout helps these models generate more general representations to perform well on inconsistent degradations.

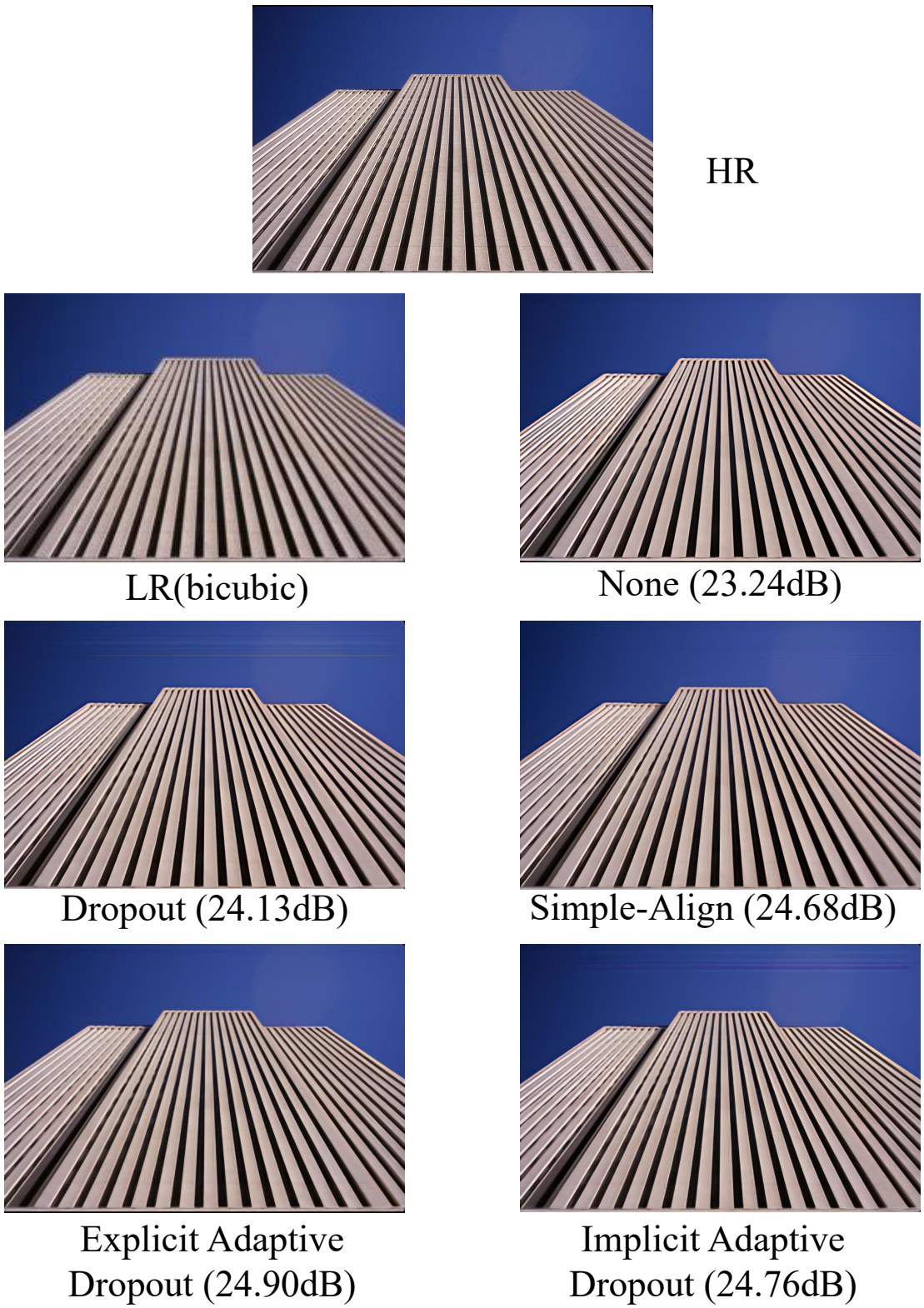

Figure 4. **Visual comparison of our methods and past methods in "bicubic".**

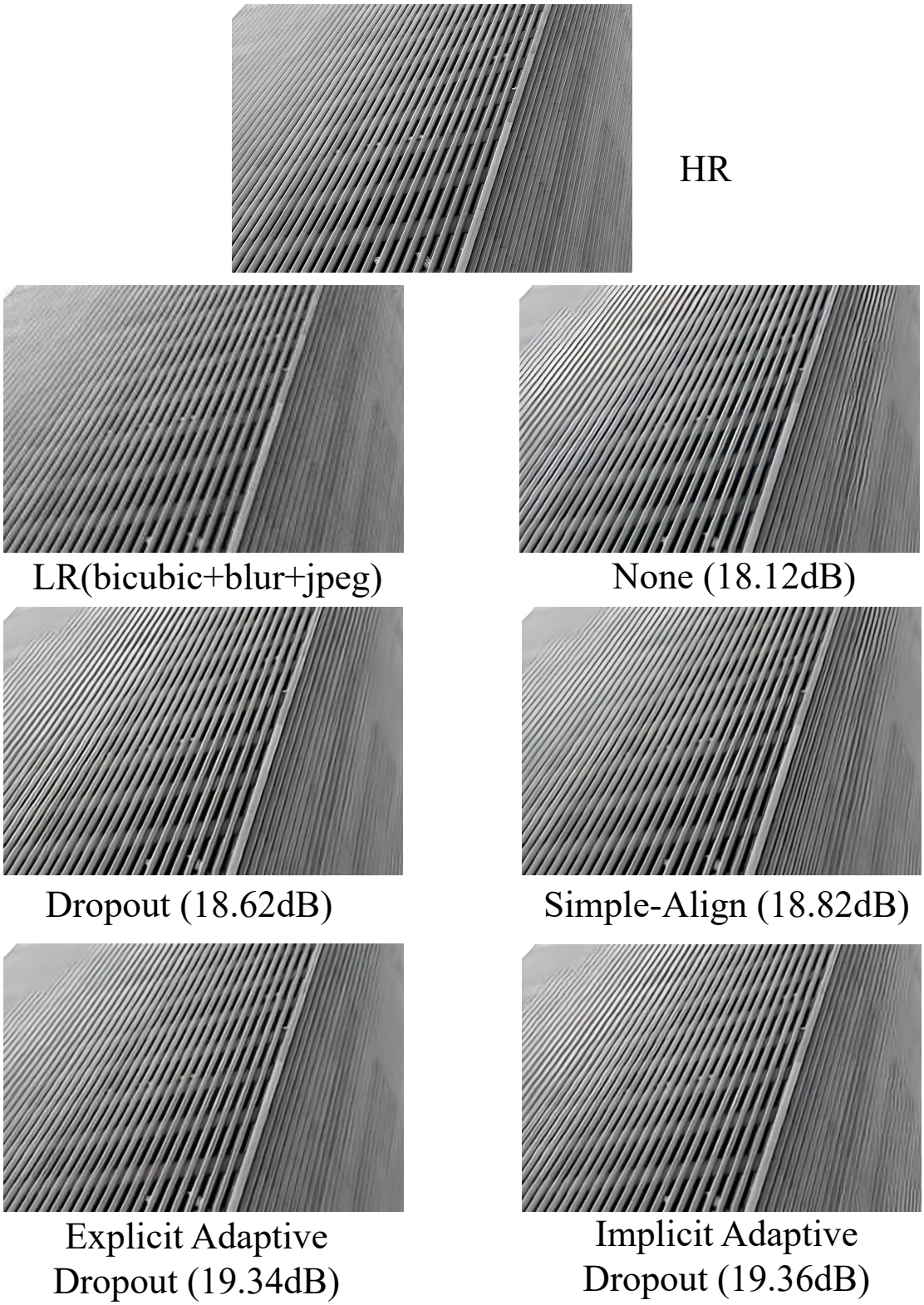

Figure 5. **Visual comparison of our methods and past methods in "bicubic+blur+jpeg".**

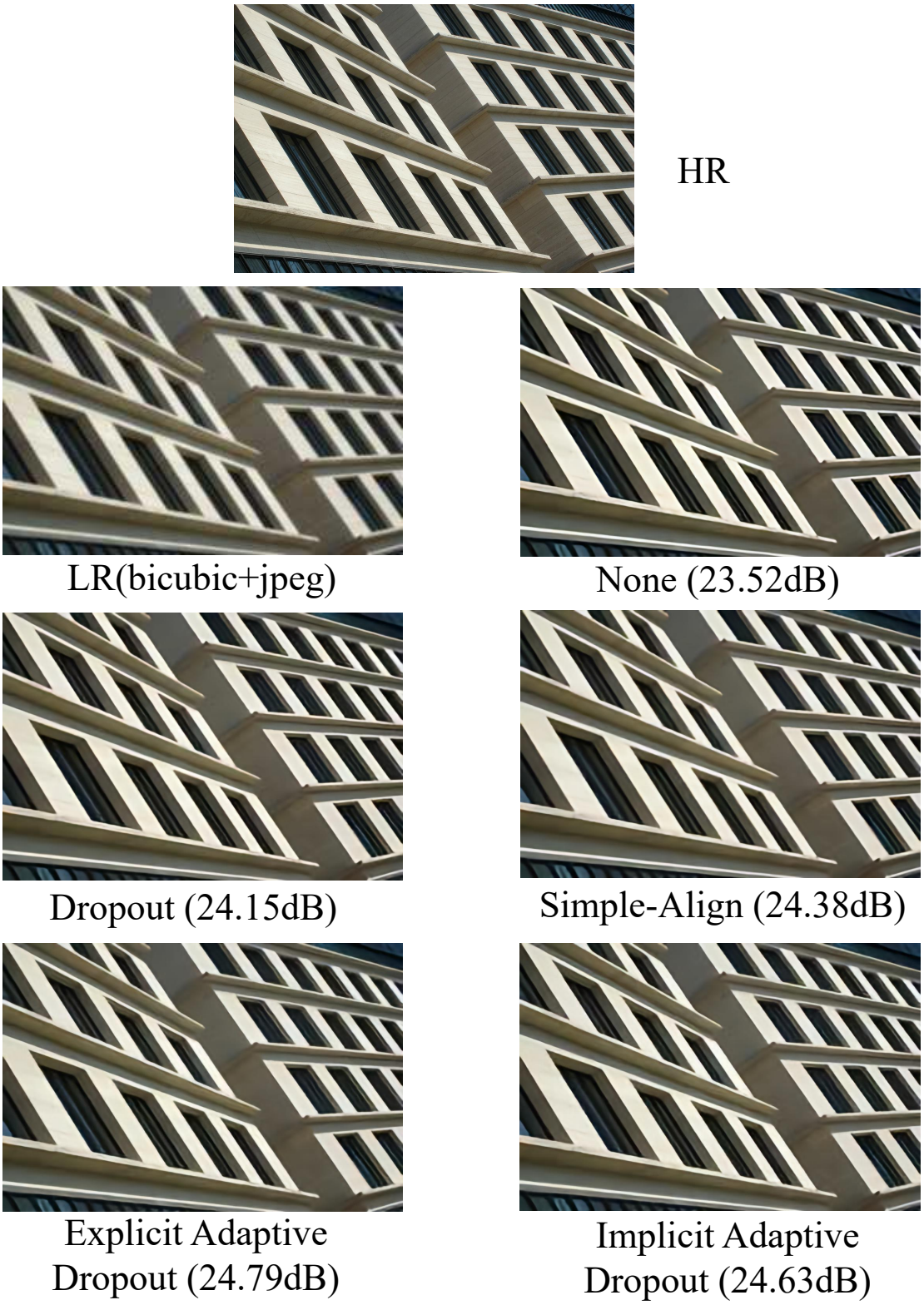

Figure 6. **Visual comparison of our methods and past methods in "bicubic+jpeg".**

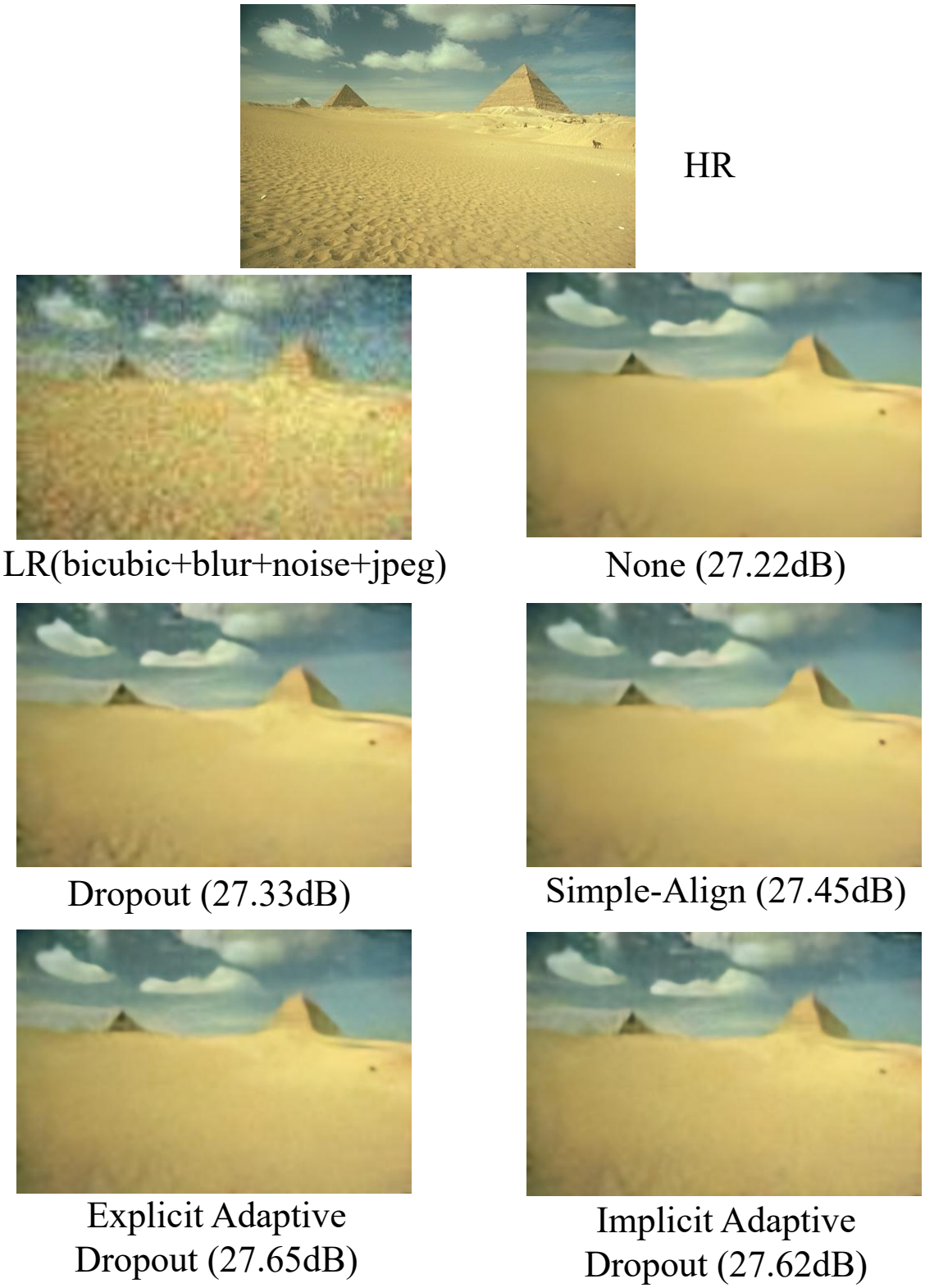

Figure 7. **Visual comparison of our methods and past methods in "bicubic+blur+noise+jpeg".**

References

[1] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. *British Machine Vision Conference*, 2012. 4

[2] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5197–5206, 2015. 4

[3] Bum Jun Kim, Hyeyeon Choi, Hyeonah Jang, Donggeon Lee, and Sang Woo Kim. How to use dropout correctly on residual networks with batch normalization. In *Uncertainty in Artificial Intelligence*, pages 1058–1067. PMLR, 2023. 3

[4] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017. 4

[5] Xiang Li, Shuo Chen, Xiaolin Hu, and Jian Yang. Understanding the disharmony between dropout and batch normalization by variance shift. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2682–2690, 2019. 3

[6] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1833–1844, 2021. 4

[7] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, pages 416–423. IEEE, 2001. 4

[8] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multimedia tools and applications*, 76:21811–21838, 2017. 4

[9] Hongjun Wang, Jiyuan Chen, Yinqiang Zheng, and Tieyong Zeng. Navigating beyond dropout: An intriguing solution towards generalizable image super resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 25532–25543, 2024. 4

[10] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE transactions on image processing*, 19(11):2861–2873, 2010. 4

[11] 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. 4