Supplemental Material: Deep Arbitrary-Scale Image Super-Resolution via Scale-Equivariance Pursuit

Xiaohang Wang[1*] Xuanhong Chen[1*] Bingbing Ni[1†] Hang Wang[2] Zhengyan Tong[1] Yutian Liu[1]
[1]Shanghai Jiao Tong University, Shanghai 200240, China [2]Huawei
{xygz2014010003,chen19910528,nibingbing}@sjtu.edu.cn

1. Overview

In this supplemental material, we first introduce our AFE operator in more detail and compare it with the previous SOTA method to draw a difference. We then give a more comprehensive introduction to CKA, the metric we use to measure feature similarity in the main text. We also perform additional experiments to determine the influence of parameters such as training iterations and the number of AFE Groups on model performance. Finally, we show more qualitative and quantitative comparison results to demonstrate the effectiveness of our method.

2. Adaptive Feature Extractor

In this section, we introduce the design of our Adaptive Feature Extractor (AFE) module in more detail and explain the configurations in our experiments. Meanwhile, we also compare with the previous SOTA scale injection method, i.e., Scale-Aware Convolution in ArbSR [13], to prove the excellent performance of our AFE module.

2.1. Algorithm of Adaptive Feature Extractor

The core of AFE is the injection of additional scale information, which assists the backbone extract features adaptively. A naive idea is to encode the magnification into a discrete space (e.g., one-hot encoding) and inject it into the model's backbone network. However, as in the problem setting of arbitrary-scale image super-resolution (ASISR), the scale can be any valid real-number (i.e., continuous and infinite). This means that these traditional discrete encoding algorithms cannot be applied to the ASISR problem. To solve this issue, we try to map the real-valued scale to a continuous high-dimensional space, and the pseudo-codes are described in Algorithm 1. Note in Algorithm 1, B is the input batch size, C is the number of channels, k denote the kernel size of convolution, dim denotes the encoding dims of sine and cosine, h and w denote the height and width of input LR images, respectively. In our experiments, we set $C = 180$, $k = 3$, and $dim = 40$.

Algorithm 1 Forward step of Adaptive Feature Extractor.

```
Input:
    Feature maps from previous stages $X_{in} \in \mathbb{R}^{B\times C\times h\times w}$; upsampling scale, $r \in \mathbb{R}$; linear mapping layer, $\mathcal{F}$; convolutional kernel basis, $\mathcal{W} \in \mathbb{R}^{C\times C\times k\times k}$; convolution bias $\mathcal{B} \in \mathbb{R}^{C}$.
Output:
    Feature maps extracted according to scales, $X_{out}$.
1: Expand the scale $r$ to a higher dimension by sine-cosine encoding and concatenate the results to obtain $r' \in \mathbb{R}^{1\times(2dim+1)}$;
2: Obtain the scale vector $v \leftarrow \mathcal{F}(r') \in \mathbb{R}^{1\times C}$;
3: Modulate the scale vector onto the convolution kernel basis to obtain the final weights $\mathcal{W}' \leftarrow v \otimes \mathcal{W}$;
4: Convolve with feature maps from previous stage. $X'_{in} \leftarrow \mathcal{W}' * X_{in}$;
5: Add the bias to get output $X_{out} \leftarrow X'_{in} + \mathcal{B}$.
6: return $X_{out} \in \mathbb{R}^{B\times C\times h\times w}$;
```

2.2. Comparison with Scale-Aware Convolution

To further illustrate the effectiveness of our AFE operator, we compare it with the previous SOTA scale injection method, Scale-Aware Convolution [13]. From the perspective of neural networks, both AFE and Scale-Aware Convolution can be regarded as dynamic convolutions. They dynamically adjust the weight of the convolution kernels by modulating a scale vector to inject the scale information into the feature extraction process explicitly, achieving the purpose of adaptive feature extraction. However, as shown in Figure 1, the implementation of these two methods has three significant differences.

First, the overall structure is different. Given a feature map X_{in}, Scale-Aware Convolution first feeds it into an hourglass module with four convolutions and a sigmoid layer to generate a guidance map M with values ranging

*Equal Contribution.
†Corresponding author: Bingbing Ni.

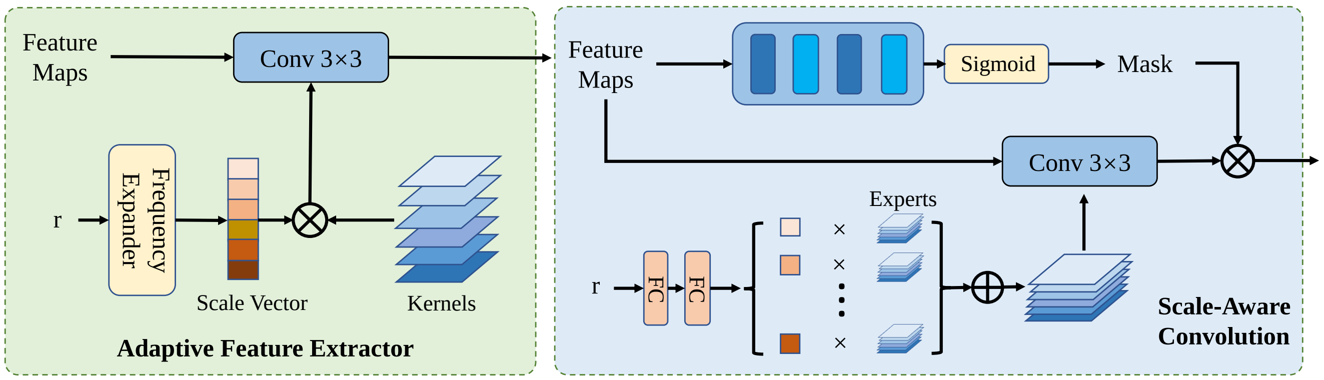

Figure 1. Comparison of AFE and Scale-Aware Convolution [13] in structure. AFE is much simpler and more flexible than the previous method.

from 0 to 1. This mask is then used to filter features extracted from the scale-aware branch, probably because the scale-aware branch will introduce some unnecessary information that needs to be filtered out. While in our design, the AFE operator abandons the branch of filtering feature maps through guidance maps and instead focuses on affecting the convolution kernel itself to enhance the equivariance of the feature extraction process. Second, the scale information affects convolution differently. In Scale-Aware Convolution, the authors pre-define a series of experts, which are actually several sets of weights, as the basic kernels. These experts will be linearly combined according to certain weights as the final convolution kernel, and the scale is used to predict their weights. Although this method can also achieve dynamic convolution, the weight of scale prediction acts on the entire expert (i.e., the kernel of one channel cannot be changed alone), which seriously limits the dynamic ability of the last convolution kernel. While in AFE, we only use one set of kernel basis, and the weights predicted by scale will be applied to each channel of the kernel. This dramatically increases the flexibility of weight combinations. Third, the complexity of AFE is significantly lower. Since Scale-Aware Convolution needs to define multiple experts, its parameter quantity is much higher than our method. When the number of experts is 4, even without considering the branch of predicting mask, the parameter amount of AFE is only about 25% of Scale-Aware Convolution.

3. Metric of Feature Similarity

As described in the main text, we use Centered Kernel Alignment (CKA) [6, 11] as a feature similarity measure. In this section, we introduce CKA in more detail.

Centered Kernel Alignment. Centered Kernel Alignment (CKA) is a representation similarity metric widely used to understand the representations learned by neural networks. In this paper, we use the Gram matrices to calculate CKA. Specifically, take $\mathbf{X} \in \mathbb{R}^{m\times p_1}$ and $\mathbf{Y} \in \mathbb{R}^{m\times p_2}$ as the two feature maps to be compared, we first compute the Gram matrices $\mathbf{K}$ and $\mathbf{T}$ as follows:

$$\mathbf{K} = \mathbf{X}\mathbf{X}^T, \mathbf{L} = \mathbf{Y}\mathbf{Y}^T. \tag{1}$$

Then we compute their normalized similarity in terms of the Hilbert-Schmidt Independence Criterion (HSIC) [3] as

$$\mathbf{K}' = \mathbf{HKH}, \mathbf{L}' = \mathbf{HLH}, \tag{2}$$

$$HSIC(\mathbf{K}, \mathbf{L}) = vec(\mathbf{K}') \cdot vec(\mathbf{L}')/(m-1)^2, \tag{3}$$

where H is a centering matrix, and $vec(\cdot)$ denotes a vectorized function. At last, CKA can be calulated as

$$CKA(\mathbf{K}, \mathbf{L}) = \frac{HSIC(\mathbf{K}, \mathbf{L})}{\sqrt{HSIC(\mathbf{K}, \mathbf{K})HSIC_0(\mathbf{L}, \mathbf{L})}}, \tag{4}$$

which is a real number in $[0, 1]$, and the higher its value, the greater the similarity.

However, the above formula is not scalable against deep architectures and large datasets. Therefore, we follow the work of [11], in which the authors propose a minibatch version that can be constructed by an unbiased estimator of the HSIC as:

$$CKA_{minibatch}(\mathbf{K}, \mathbf{L}) = \frac{\frac{1}{k}\sum_k^{i=1} HSIC(\mathbf{X_i X_i^T}, \mathbf{Y_i Y_i^T})}{\sqrt{\sum_k^{i=1} HSIC(\mathbf{X_i X_i^T}, \mathbf{X_i X_i^T}) \sum_k^{i=1} HSIC(\mathbf{Y_i Y_i^T}, \mathbf{Y_i Y_i^T}))}}. \tag{5}$$

Experiment Details. In our experiments, we choose batchsize=4 and compute the CKA similarity between models for different scales on the Urban100 dataset. For ArbSR [13], we compare the feature maps from each ResBlock; for HAT [1] and our EQNet, we compare the feature generated by each attention layer.

Table 1. Quantitative comparison (PSNR) for different number of AFE Groups and WSA Blocks per group. The best and second-best results are marked in red and blue colors, respectively. We choose Model IV as the final model to report in the main text.

	num of Groups	WSAs per Group	paras	Urban100 [5]		
				×2	×4	×*6
I	4	4	8.5M	33.35	26.92	24.24
II	4	5	9.6M	33.41	27.17	24.50
III	5	5	11.6M	33.52	27.22	24.57
IV	6	5	13.6M	33.62	27.30	24.66
V	6	6	15.2M	33.64	27.31	24.63

Table 2. Quantitative comparison (PSNR) for different training stages of our EQNet. The best and second-best results are marked in red and blue colors, respectively. Note that we enable the **pre-training strategy** for this comparison.

	Iterations		Urban100 [5]		
	ImageNet	DF2K	×2	×4	×*6
I	200k	0	33.00	26.92	24.36
II	400k	0	33.29	27.28	24.67
III	600k	0	33.41	27.47	24.82
IV	800k	0	33.45	27.52	24.89
V	800k	200k	33.83	27.54	24.83

4. Additional Experiments

In this section, we conduct additional experiments to illustrate the role of different hyperparameters in the model.

4.1. The Number of Groups and Blocks

Generally speaking, a larger model usually has a larger capacity and more vital learning ability. However, such a model also commonly suffers from problems such as being difficult to train and prone to overfitting due to a large number of parameters. Therefore, it is important to make a trade-off on model size. To this end, in this subsection, we change the number of AFE Groups and the number of window-based self-attention (WSA) blocks per group in our backbone and test their performance on the Urban100 dataset. The results at scales $\times 2/4/6$ are shown in Table 1. Note that we DO NOT adopt the pre-training strategy in these experiments.

From Table 1, we observe that from Model I to Model IV, as the model size becomes larger, the performance both in training distribution and out of the distribution continues to improve. However, comparing Model IV and Model V, it can be observed that the benefit of increasing the parameters is not as evident as before. The performance is even degraded in the case of $\times 6$. This means that Model V is likely to be overfitting. Therefore, we choose Model IV as the final model in the main text.

4.2. Training Iterations

We also explore the performance of EQNet at different stages of the training process, and the results in PSNR are shown in Table 2. Note that we use the pre-training strategy for this comparison. We first train EQNet on ImageNet dataset [7] for 800k iterations, and then train on DF2K dataset [12] for the rest 200k iterations.

From Table 2, it can be observed that with the increase of training iterations, our model's performance continues to improve. We also observe that cases out of the training distribution converge faster than those inside the distribution. This means that from the middle of the training process (about 600k iterations), EQNet already has a basic certainty of equivariance and generalization. On the other hand, at scale $\times 2$, there is a significant gap (0.38dB) between Model IV and Model V, while at $\times 4$ and $\times 6$, the gaps are much smaller, indicating that the DF2K dataset is more beneficial for training at lower scales. As a matter of fact, arbitrary-scale image super-resolution can be viewed as a muti-task problem, which is much more complicated than normal super-resolution. Therefore, training the network for a long time is essential to fully exploit the model's potential.

4.3. More Comparisons

In this subsection, we conduct more qualitative and quantitative comparisons of our EQNet with previous SOTA methods.

Quantitative Comparison. To further test the performance of our method, we continue to compare our EQNet model with other state-of-the-art arbitrary-scale SR methods: MetaSR [4], ArbSR [13], LIIF [2] and LTE [8]. In addition to the average Peak Signal to Noise Ratio (PSNR), we also report these models' Structural Similarity (SSIM). The quantitative results for $\times 2.25/2.75/3.25/3.75/4.25/4.75/5.25$ SR are shown in Table 3. Note that all comparison ASISR models are based on one single model protocol. It can be observed that our method achieves the best results under most configurations in terms of both PSNR and SSIM, especially at out-of-distribution scales $\times 4.25/4.75/5.25$.

Qualitative Comparison. We also conduct more qualitative comparisons of different methods with different scale factors. Figure 2 shows the results on Urban100 dataset [5]. It can be observed that EQNet is far superior to other methods in processing complex textures, which further illustrates our method's ability to restore high-frequency information. Figure 3 displays the results on BSD100 dataset [9]. The results show that our method also has significant advantages in reconstructing details when processing natural images (for example, the edge contour of the window in the first row). Figure 4 shows the results on Manga109 dataset [10]. From the comparison, we observe that our model is able to restore more clear edges for printed texts at different scales, such as the letters E, A, P, R, and so on. These comparisons further demonstrate the effectiveness of our equivariant model.

References

[1] Xiangyu Chen, Xintao Wang, Jiantao Zhou, and Chao Dong. Activating more pixels in image super-resolution transformer. *arXiv preprint arXiv:2205.04437*, 2022. 2

[2] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8628–8638, 2021. 3, 5

[3] Arthur Gretton, Kenji Fukumizu, Choon Teo, Le Song, Bernhard Schölkopf, and Alex Smola. A kernel statistical test of independence. *Advances in neural information processing systems*, 20, 2007. 2

[4] Xuecai Hu, Haoyuan Mu, Xiangyu Zhang, Zilei Wang, Tieniu Tan, and Jian Sun. Meta-sr: A magnification-arbitrary network for super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1575–1584, 2019. 3, 5

[5] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, pages 5197–5206, 2015. 3, 6

[6] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2661–2671, 2019. 2

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 3

[8] Jaewon Lee and Kyong Hwan Jin. Local texture estimator for implicit representation function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1929–1938, 2022. 3, 5

[9] 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 *ICCV*, volume 2, pages 416–423. IEEE, 2001. 3, 6

[10] 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(20):21811–21838, 2017. 3, 7

[11] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020. 2

[12] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. Ntire 2017 challenge on single image super-resolution: Methods and results. In *CVPR workshops*, pages 114–125, 2017. 3

[13] Longguang Wang, Yingqian Wang, Zaiping Lin, Jungang Yang, Wei An, and Yulan Guo. Learning a single network for scale-arbitrary super-resolution. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4801–4810, 2021. 1, 2, 3, 5

Table 3. Quantitative comparison (in average PSNR and SSIM) for **arbitrary-scale SR** with state-of-the-art methods on benchmark datasets. The best and second-best results are marked in red and blue colors, respectively. "†" indicates that methods adopt pre-training strategy on ImageNet.

Method	Scale	Set5		Set14		BSD100		Urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic		32.99	0.9192	29.66	0.8494	28.85	0.8190	26.21	0.8152	29.81	0.9177
MetaSR [4]		37.01	0.9513	32.80	0.9002	31.20	0.8753	31.71	0.9211	37.75	0.9674
ArbSR [13]		37.02	0.9537	32.84	0.9017	31.24	0.8767	31.71	0.9196	37.84	0.9710
LIIF [2]	×2.25	37.05	0.9528	32.84	0.9023	31.27	0.8761	31.54	0.9178	37.72	0.9709
LTE [8]		37.20	0.9539	33.12	0.9235	31.42	0.8796	32.08	0.9211	38.11	0.9707
Ours		27.18	0.9530	33.25	0.9253	31.37	0.8779	32.20	0.9245	28.14	0.9714
Ours†		37.22	0.9541	33.32	0.9286	31.43	0.8792	32.57	0.9286	38.19	0.9726
Bicubic		31.06	0.8864	28.29	0.8002	27.71	0.7728	24.98	0.7610	27.64	0.8774
MetaSR [4]		35.36	0.9380	31.19	0.8627	29.86	0.8345	29.73	0.8849	35.46	0.9543
ArbSR [13]		35.39	0.9374	31.23	0.8638	29.91	0.8385	29.71	0.8856	35.51	0.9567
LIIF [2]	×2.75	35.38	0.9372	31.11	0.8638	29.83	0.8306	29.57	0.8829	35.27	0.9562
LTE [8]		35.51	0.9385	31.44	0.8690	29.89	0.8331	29.84	0.8895	35.85	0.9586
Ours		35.52	0.9381	31.47	0.8682	29.91	0.8335	30.12	0.8916	35.77	0.9570
Ours†		35.55	0.9389	31.60	0.8689	29.96	0.8343	30.58	0.8977	35.86	0.9593
Bicubic		29.21	0.8528	27.17	0.7569	26.97	0.7339	24.09	0.7142	26.40	0.8375
MetaSR [4]		33.98	0.9177	29.73	0.8283	28.99	0.8042	28.19	0.8505	33.49	0.9398
ArbSR [13]		34.03	0.9226	29.87	0.8290	29.00	0.8038	28.22	0.8518	33.55	0.9410
LIIF [2]	×3.25	34.12	0.9221	30.00	0.8302	28.83	0.7914	28.19	0.8500	33.35	0.9406
LTE [8]		34.42	0.9239	30.26	0.8324	28.87	0.7946	28.73	0.8598	33.76	0.9423
Ours		34.37	0.9235	30.30	0.8339	28.91	0.7952	28.85	0.8628	33.89	0.9425
Ours†		34.44	0.9247	30.43	0.8365	28.97	0.7960	29.12	0.8683	34.16	0.9457
Bicubic		28.98	0.8331	26.37	0.7224	26.20	0.6885	23.30	0.6756	25.41	0.8043
MetaSR [4]		33.15	0.9139	29.05	0.8025	27.98	0.7583	27.07	0.8180	31.90	0.9222
ArbSR [13]		33.12	0.9126	29.01	0.8011	28.01	0.7605	27.09	0.8192	31.93	0.9247
LIIF [2]	×3.75	33.04	0.9073	29.14	0.8004	28.07	0.7575	27.13	0.8190	31.83	0.9251
LTE [8]		33.10	0.9083	29.22	0.8039	28.10	0.7606	27.65	0.8324	32.41	0.9303
Ours		33.13	0.9092	29.27	0.8045	28.16	0.7613	27.69	0.8337	33.51	0.9298
Ours†		33.30	0.9110	29.52	0.8071	28.21	0.7627	28.01	0.8402	32.70	0.9323
Bicubic		28.08	0.7983	24.17	0.6945	25.79	0.6697	22.95	0.6442	24.49	0.7698
MetaSR [4]		31.75	0.8890	28.47	0.7765	27.53	0.7427	26.13	0.7872	30.29	0.9035
ArbSR [13]		31.76	0.8887	28.46	0.7759	27.52	0.7420	26.10	0.7864	30.31	0.9044
LIIF [2]	×*4.25	32.05	0.8917	28.50	0.7748	27.46	0.7286	26.25	0.7896	30.52	0.9094
LTE [8]		32.43	0.8952	28.55	0.7774	27.46	0.7314	26.67	0.8044	30.66	0.9112
Ours		32.30	0.8932	28.72	0.7795	27.43	0.7317	26.79	0.8068	30.87	0.9135
Ours†		32.42	0.8966	28.78	0.7807	27.61	0.7341	27.11	0.8134	31.41	0.9183
Bicubic		27.37	0.7717	23.87	0.6627	24.25	0.6401	22.47	0.6133	23.85	0.7435
MetaSR [4]		30.62	0.8716	27.71	0.7439	26.92	0.7091	25.36	0.7549	29.06	0.8861
ArbSR [13]		30.59	0.8704	27.67	0.7436	26.89	0.7085	25.21	0.7512	28.88	0.8820
LIIF [2]	×*4.75	31.15	0.8762	27.88	0.7522	26.96	0.7044	25.55	0.7626	29.47	0.8938
LTE [8]		31.40	0.8802	27.87	0.7511	26.98	0.7078	25.80	0.7761	29.79	0.8977
Ours		31.46	0.8794	28.01	0.7553	27.03	0.7085	25.92	0.7794	29.95	0.9003
Ours†		31.67	0.8831	28.12	0.7580	27.10	0.7099	26.36	0.7879	30.35	0.9045
Bicubic		26.91	0.7579	23.23	0.6437	23.90	0.6096	22.09	0.5885	21.83	0.7201
MetaSR [4]		29.95	0.8583	27.10	0.7285	26.27	0.6761	24.73	0.7303	28.19	0.8654
ArbSR [13]		29.86	0.8558	26.89	0.7262	26.22	0.6742	24.47	0.7190	27.59	0.8570
LIIF [2]	×*5.25	30.38	0.8611	27.38	0.7310	26.52	0.6824	24.95	0.7374	28.54	0.8779
LTE [8]		30.51	0.8652	27.46	0.7353	26.56	0.6862	25.19	0.7514	28.96	0.8803
Ours		30.56	0.8660	27.49	0.7366	26.57	0.6866	25.32	0.7546	29.14	0.8845
Ours†		30.87	0.8687	27.63	0.7379	26.67	0.6884	25.70	0.7634	29.36	0.8895

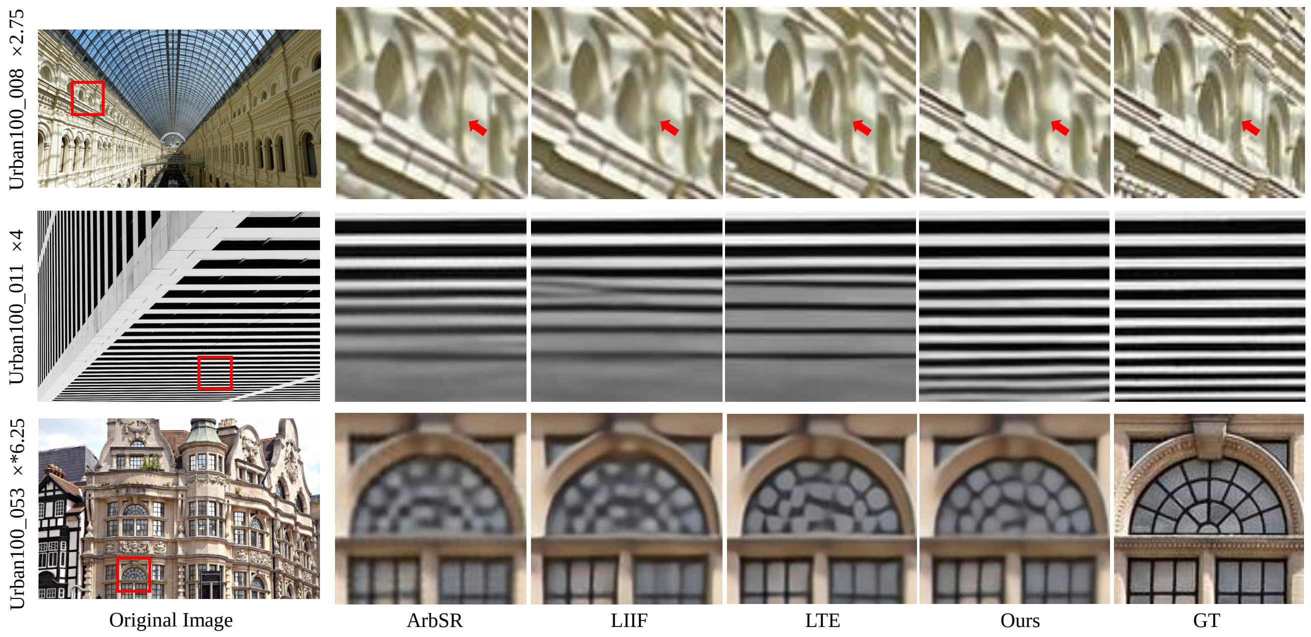

Figure 2. Visual comparison for arbitrary-scale SR models on Urban100 dataset [5]. In the first row, our method can restore a more realistic shape of the hole. In the second row, our method reconstructs dense textures accurately, while other methods produce blurred results. The third row shows that our model can keep the original contour of the object as much as possible when the scale is large. This further illustrates that our method's ability to restore high-frequency information.

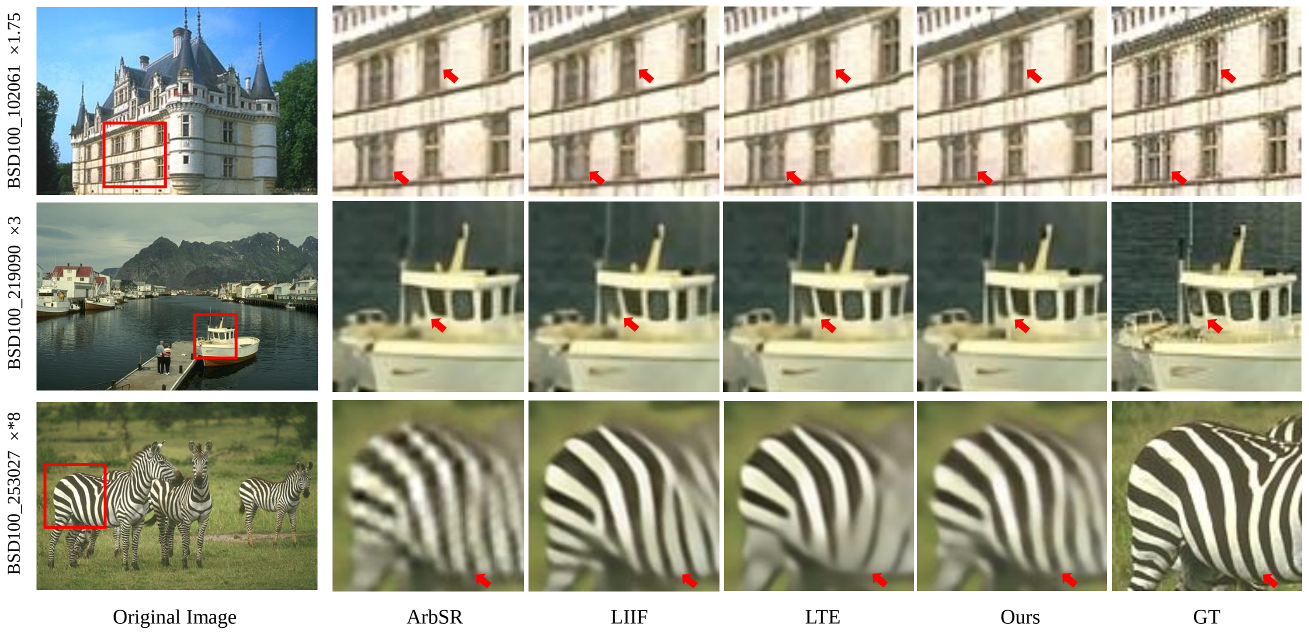

Figure 3. Visual comparison for arbitrary-scale SR models on BSD100 [9] dataset. The results show that our method also has significant advantages in reconstructing details when processing natural images (for example, the edge contour of the window in the first row and second row).

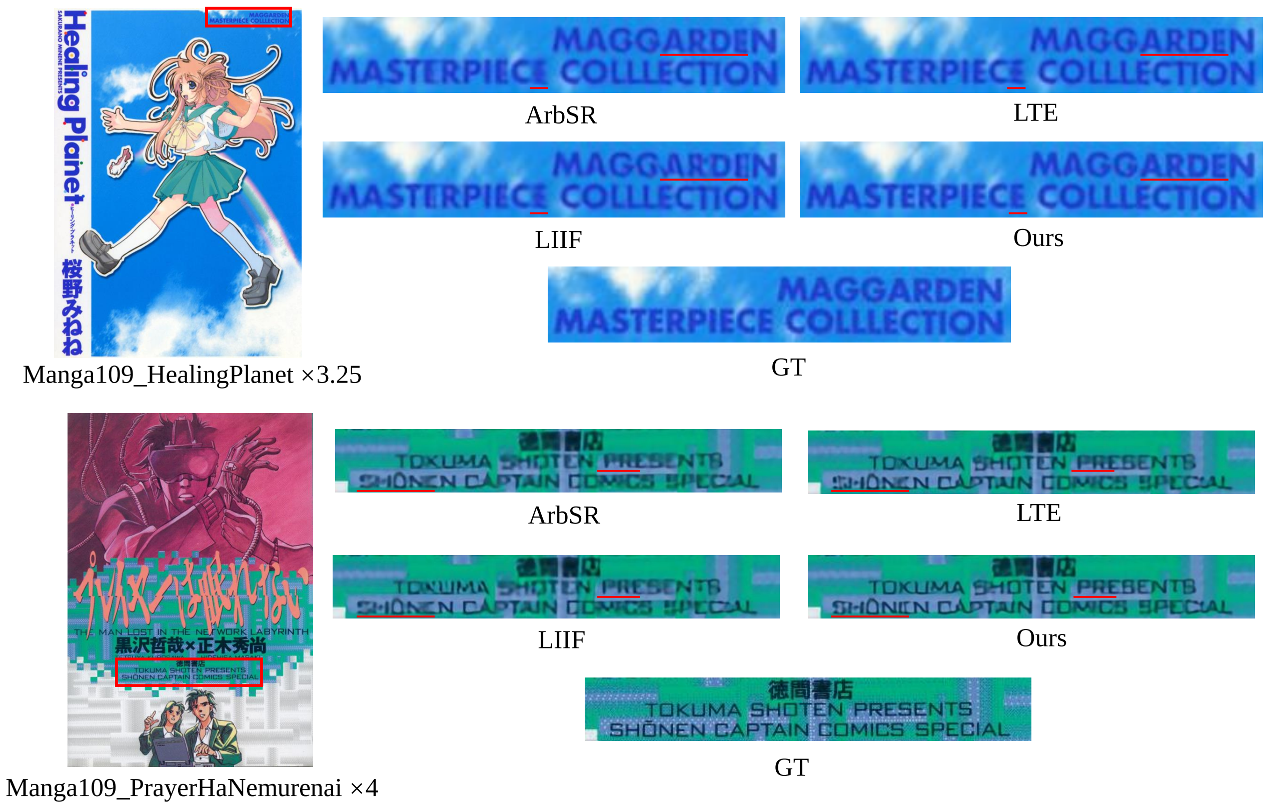

Figure 4. Visual comparison for arbitrary-scale SR models on Manga109 [10] dataset. It can be observed that our model is able to restore more clear edges for printed texts at different scales, such as the letters E, A, P, R, and so on.