Activating More Information in Arbitrary-Scale Image Super-Resolution

Yaoqian Zhao, Qizhi Teng, *Member, IEEE*, Honggang Chen, *Member, IEEE*, Shujiang Zhang, Xiaohai He, *Member, IEEE*, Yi Li, Ray E. Sheriff, *Senior Member, IEEE*

***Abstract*—Single-image super-resolution (SISR) has experienced vigorous growth with the rapid development of deep learning. However, handling arbitrary scales (*e.g.,* integers, non-integers, or asymmetric) using a single model remains a challenging task. Existing super-resolution (SR) networks commonly employ static convolutions during feature extraction, which cannot effectively perceive changes in scales. Moreover, these continuous-scale upsampling modules only utilize the scale factors, without considering the diversity of local features. To activate more information for better reconstruction, two plug-in and compatible modules for fixed-scale networks are designed to perform arbitrary-scale SR tasks. Firstly, we design a Scale-aware Local Feature Adaptation Module (SLFAM), which adaptively adjusts the attention weights of dynamic filters based on the local features and scales. It enables the network to possess stronger representation capabilities. Then we propose a Local Feature Adaptation Upsampling Module (LFAUM), which combines scales and local features to perform arbitrary-scale reconstruction. It allows the upsampling to adapt to local structures. Besides, deformable convolution is utilized letting more information to be activated in the reconstruction, enabling the network to better adapt to the texture features. Extensive experiments on various benchmark datasets demonstrate that integrating the proposed modules into a fixed-scale SR network enables it to achieve satisfactory results with non-integer or asymmetric scales while maintaining advanced performance with integer scales.**

This work was supported in part by the National Natural Science Foundation of China (No. 62001316), in part by the Research Fund of Guangxi Key Lab of Multi-source Information Mining & Security (No. MIMS22-14), in part by the Opening Foundation of Key Laboratory of Computer Vision and System, Ministry of Education, Tianjin University of Technology, China (No. TJUT-CVS20220001), in part by the Open Foundation of Yunnan Key Laboratory of Software Engineering under Grant (No. 2023SE206), in part by the Opening Foundation of Zhejiang Intelligent Transportation Engineering Technology Research Center, China (No. 2023ERCITZJ-KF15), and in part by the Fundamental Research Funds for the Central Universities (No. SCU2023D062 and No. 2022CDSN-15-SCU). (Corresponding author: Honggang Chen.)

Yaoqian Zhao is with the College of Electronics and Information Engineering, Sichuan University, Chengdu 610065, China, and also with Guangxi Key Lab of Multi-source Information Mining & Security, Guangxi Normal University, Guilin 541004, China (email: 244716386@qq.com).

Qizhi Teng and Xiaohai He are with the College of Electronics and Information Engineering, Sichuan University, Chengdu 610065, China (e-mail: qzteng@scu.edu.cn; hxh@scu.edu.cn).

Honggang Chen is with the College of Electronics and Information Engineering, Sichuan University, Chengdu 610065, China, also with the Key Laboratory of Computer Vision and System, Ministry of Education, Tianjin University of Technology, Tianjin 300384, China, and also with the Yunnan Key Laboratory of Software Engineering, Yunan University, Kunming 650600, China (e-mail: honggang_chen@scu.edu.cn).

Shujiang Zhang is with the Zhejiang Intelligent Transportation Engineering Technology Research Center Enjoyor Technology Co., Ltd., Hangzhou 311400, China (e-mail: 34209310@qq.com).

Yi Li is with the D.I. Sinma (Sichuan) Machinery Co., Ltd., Suining, China (e-mail: 943787101@qq.com).

Ray E. Sheriff is with the Department of Computer Science, Edge Hill University, Ormskirk L394QP, United Kingdom (e-mail: Sheriffr@edgehill.ac.uk).

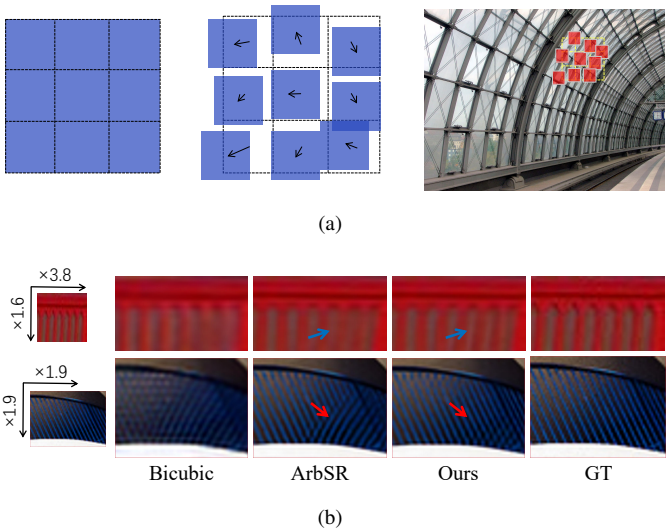


Fig. 1. Activating more information makes the reconstructed image better adapt to the content of the image. (a) The receptive field is deformed based on the texture of the image. (b) Comparison of visual results achieved by Bicubic, ArbSR [8] and proposed model for asymmetric scale factors and non-integer scale factors.

***Index Terms*—Super-resolution, arbitrary-scale, scale-aware, local feature adaptation, dynamic convolution, deformable convolution.**

I. Introduction

SINGLE image super-resolution (SISR) aims to restore a high-resolution (HR) image from its low-resolution (LR) version. It is a challenging ill-posed issue and also a long-lasting fundamental task in the field of computer vision. The emergence of deep learning has led to the proposal of convolutional neural network (CNN)-based image SR methods [1]–[7], which have shown remarkable superiority.

In real-world scenarios such as medical imaging [9], [10], remote sensing [11], [12], and security surveillance [13], [14], it is often necessary to upscale LR images to multiple different scales, including non-integer and asymmetric, because images at different scales exhibit distinct details. However, most SR methods do not support arbitrary-scale reconstruction in a single model. If there is a need to upscale images to different scales, multiple models would have to be retrained, which is clearly a wasteful use of resources.

To address this issue, there have been some methods proposed that support multiple scales and arbitrary scales. The first category includes methods that support multiple

integer scale factors in a single model, such as MDSR [15], using multiple branches to handle multi-scale tasks. However, MDSR [15] can only deal with integer scale factors after training. The second category includes methods that can achieve arbitrary-scale SR. MetaSR [16] is a groundbreaking work in continuous-scale SR, where it predicts the upsampling weights dynamically for different scale factors using a fully connected (FC) neural network in the upsampling process through meta-learning [17]. MetaSR [16] has achieved good performance on arbitrary-scale SR tasks. However, it only utilizes scale information in the upsampling process, and the features in the backbones are the same regardless of the scale, which obviously requires improvement. Additionally, MetaSR [16] cannot handle asymmetric scale factors.

ArbSR [8] introduces the use of scale information in the feature extraction of the backbone network and can handle non-integer and asymmetric scale factors concurrently. However, in the upsampling process, ArbSR [8] only utilizes scale factors and position information to generate dynamic convolutional kernels, where the filter weights are position-dependent. This results in a significant drawback: when the sizes of different input images and the scale factors are the same, the corresponding position's upsampling filter weights are the same, which is clearly not optimal. For the same position in different images, since the pixel values vary, the filter weights should also vary accordingly. For example, the filter weights in the edge regions should be different from those in the smooth regions. Therefore, it is necessary to incorporate local image features to collaboratively generate dynamic upsampling convolutional kernels.

These methods use fixed receptive fields for reconstruction during the upsampling process, which means pixels at different positions refer to information only from their fixed square-shaped local regions. However, due to the diversity of image content, dynamically determining the receptive field based on texture features makes the network highly adaptive to the image content and improves its ability to focus on pertinent image regions, as shown in Fig. 1(a). This requires learning additional offset information. Therefore, we introduce deformable convolution to satisfy this requirement.

To activate more information, we propose two plug-in modules that enable a fixed-scale network to achieve arbitrary-scale SR in a single model, making the reconstruction process more adaptive to the local structure of the image. The modules we developed are the Scale-aware Local Feature Adaptation Module (SLFAM) and the Local Feature Adaptation Upsampling Module (LFAUM). In order to enable the network to adapt to scale factors and local image features during the feature extraction process, we introduce the SLFAM based on dynamic convolution [18]. This module is capable of generating dynamic scale-aware local feature adaptive filters, allowing the network to extract image features at different scales and regions. To achieve arbitrary-scale SR, we propose the LFAUM based on deformable convolution [19], [20]. It enables the receptive field to adaptively change its range and direction during the image reconstruction process, enhancing the network's ability to model geometric transformations and generating SR images that are more adaptable to the texture of the image.

The proposed modules have low computational cost and strong adaptability, making it easy to integrate them into existing state-of-the-art (SOTA) SR methods. By inserting SLFAM into the backbone network and replacing the upsampling module at the end of the network with our LFAUM, we can transform the original fixed-scale SR network into a continuous-scale SR network. Equipped with our modules, the fixed-scale SR network can generate super-resolved images with clear texture features at asymmetric and non-integer scale factors, while keeping advanced performance at integer scales, as shown in Fig. 1(b).

Our major contributions in this work are as follows:

1) We propose two efficient plug-in modules that can be combined with existing fixed-scale networks to achieve arbitrary-scale SR tasks.
2) SLFAM is developed to address the lack of adaptability to scale and local structures in the feature extraction process by generating dynamic adaptive convolutional kernels.
3) LFAUM is designed to handle arbitrary-scale reconstruction tasks and achieve multiple information fusion during the upsampling process.
4) Extensive experimental results illustrate the effectiveness of the proposed method, which significantly improves the baseline results. Both objective metrics and visual quality outperform SOTA arbitrary-scale SR methods.

The remainder of this paper is organized as follows. Section II introduces the related works. We propose two efficient plug-in modules for arbitrary-scale SR tasks in Section III. In Section IV, experiments and discussions are presented to illustrate the performance of the proposed modules.Finally, we conclude this paper in Section V.

II. Related Works

In this section, we will briefly review fixed-scale single image super-resolution (SISR) algorithms in the field of deep learning and then discuss some related works on arbitrary-scale SR.

A. Single Image Super-Resolution

Existing SISR algorithms can be grouped into three main categories: interpolation-based [21]–[23], reconstruction-based [24]–[26], and learning-based [27]–[32]. Early learning-based methods, such as exemplar or dictionary-based approaches, rely on external image databases to generate HR images by transferring relevant patches from the database images.

Deep neural networks can learn end-to-end mappings from LR space to HR space. Due to their powerful feature representation and model fitting capabilities, CNN-based SR methods [33], [34] show significant advantages over traditional methods. CNN-based algorithms are broadly categorized into two types: the pre-upsampling SR model and the post-upsampling SR model. The former one, SRCNN [1] for example, upsamples the LR image to the demanded size firstly and then uses a CNN to recover high-frequency details. VDSR [35] introduces residual learning for SR model training. However,

pre-upsampling leads to subsequent feature extraction operations being performed in a high-dimensional space, resulting in time-consuming and high computational costs processes.

To address this issue, researchers proposed post-upsampling SR methods to enhance computational efficiency, which firstly study the low-dimensional space features of LR images and then connect a learnable reconstruction layer to upsample the features. Lim *et al.* [36] introduce EDSR, a very deep and wide network, which trains large-scale SR models by utilizing residual scaling techniques and removing batch normalization layers. RDN [37] propose dense feature fusion for image SR. RCAN [38] and SAN [39] respectively introduce channel attention mechanisms and second-order channel attention to enhance SR performance. HAT [40] combines channel attention [41] and window-based self-attention mechanisms, leveraging their complementary advantages in global statistical information and local fitting capabilities, further improving SR performance. Post-upsampling methods become the commonly used frameworks in the field of SR because of the fewer resources consumed in training and inference processes. Many of these methods utilize an upsampling module called the PixelShuffle convolution layer, which is introduced by Shi *et al.* [42]. This layer generates multiple channels through convolution and then integrates the pixels of each channel to achieve upsampling functionality. The deconvolution layer [43] is another end-to-end learnable reconstruction layer, which achieves upsampling by performing the operation of padding and interpolation on the feature map using the inverse convolution kernel.

While these SISR methods achieve promising results, they all require training of separate models for specific integer scale factors, such as $\times 2$, $\times 3$, or $\times 4$, and they cannot achieve SR with fractional scale factors. Considering the limitations of memory and computational resources, this clearly restricts the practical deployment of these models.

B. Arbitrary-Scale SR

In practical scenarios, users often need to upscale images to arbitrary sizes. To address this issue, Lim *et al.* [15] propose the MDSR model, which consists of multiple upsampling branches designed for different scale factors. Unfortunately, MDSR [15] cannot effectively handle fractional scale factors for SR.

Inspired by meta-learning [17], Hu *et al.* [16] propose MetaSR, which uses the scale factors as input to predict the weights of the upsampling filters, enabling arbitrary-scale reconstruction in a single model. However, MetaSR [16] does not utilize scale information within the backbone network, resulting in the absence of differentiated features for images of different scales during the feature extraction.

To fully leverage scale information, Fu *et al.* [44] propose RSAN, which utilizes scale information as prior knowledge to learn discriminative features, with the aim to achieve better performance. To learn continuous image representations, LIIF [45] trains the network with implicit neural representations. LTE [46] introduces an advantage frequency estimator based on LIIF [45], which improves the performance.

However, the aforementioned methods are unable to achieve SR images with asymmetric scale factors. To address the issue, Hu *et al.* [8] propose ArbSR [8], which utilizes conditional convolution to generate dynamic scale-aware kernels, enabling SR for images with asymmetric and non-integer scale factors. This method predicts the upsampling filters weights based solely on scale factors and coordinate information. However, when the input image sizes and upscaling factors are the same, the corresponding weights of the upsampling filters at different positions are identical, which clearly overlooks the influence of content on the upsampling filters.

In order to fully utilize the information of the content, the weights of the upsampling convolutional kernels should not only be position-dependent but also content-dependent. Dynamic convolutional kernels generated by combining scales and local feature information can reconstruct images with richer details and finer textures.

III. Methodology

To solve the problem of arbitrary-scale SR, we propose two plug-in modules, the Scale-aware Local Feature Adaptation Module (SLFAM) and the Local Feature Adaptation Upsampling Module (LFAUM). Plug-in modules are more flexible and adaptable than a whole network, which can easily be combined with a fixed-scale network, such as EDSR [36], RDN [37] and RCAN [38], to generate arbitrary-scale SR results. This section mainly introduces the details of our implementation.

A. Overall Structure

The overall network structure of our arbitrary-scale SR algorithm is demonstrated in Fig. 2. The network is divided into three sections: shallow feature extraction, deep feature extraction, and arbitrary-scale upsampling. Firstly represent the LR image and the paired ground truth as I_{LR} and I_{HR}, respectively. SR task is aimed at generating an image I_{SR} with the identical spatial resolution as I_{HR}. Then, I_{LR} is processed through shallow feature extraction to obtain the feature F_0:

$$F_0 = f_0(I_{LR}) \tag{1}$$

where $f_0(\cdot)$ represents the shallow feature extraction using a 3x3 convolutional layer.

The deep feature extraction module takes the shallow feature F_0 and two scale factors, r_h and r_w in the horizontal and vertical directions, as inputs. It consists of multiple residual groups (RG) that generate the deep feature F_1. The proposed SLFAM is plugged into the deep feature extraction, with each RG followed by a SLFAM. Therefore, the obtained deep feature F_1 can be represented as:

$$F_1 = f_n(...(f_2(f_1(F_0, r_h, r_w)))) \tag{2}$$

where $f_i(\cdot)$ (for i ranging from 1 to n) represents the i-th backbone RG module and a SLFAM connected in series.

As for upsampling, the proposed LFAUM enables arbitrary-scale SR in a single model. To make the network adapt to scale factors and local features, the upsampling process takes the deep feature F_1, along with the scale factors r_h and r_w as

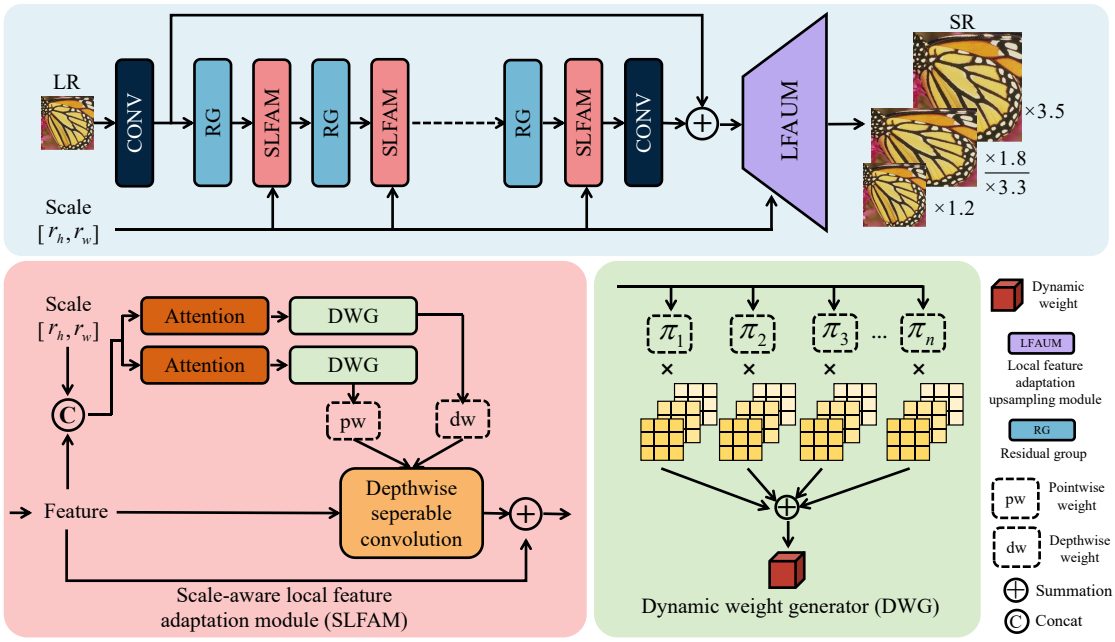


Fig. 2. Architectures of proposed networks. The details of LFAUM are shown in Fig. 3.

inputs. After being processed by LFAUM, the final SR image I_{SR} is obtained:

$$I_{SR} = f_u(F_1, r_h, r_w) \tag{3}$$

Here, $f_u(\cdot)$ represents the reconstruction process performed by the LFAUM.

Regarding the training optimization of the model, the L_1 loss is adopted as the loss function for the SR network, defined as follows:

$$L(\theta) = ||I_{SR} - I_{HR}||_1 \tag{4}$$

where θ represents the parameters of the SR network, I_{SR} represents the reconstructed HR images generated by the network, and I_{HR} represents the corresponding ground truth HR images.

B. Scale-Aware Local Feature Adaptation

For different scale factors, conventional SR methods usually require training separate networks to extract LR features specific to each scale factor, where each scale factor of the SR task has its own corresponding network weights. However, training separate networks for each scale factor is not feasible under the condition of arbitrary-scale image SR, especially for non-integer scale factors. Therefore, there is a need to train a single model to address arbitrary-scale problems. Although SR tasks with different scale factors share similarities, the fundamental differences exist, particularly when there is a significant difference in scale factors. Extracting features with specific scales is the vital aspects to accomplish arbitrary-scale SR in a single model.

If we only use the feature extraction network of the backbone network to extract features, the weights of the feature extraction network will be fixed for different scale factors, resulting in the extraction of the same features. Obviously, this is unsatisfactory; for different scale factors, the extracted features should be different. As the scale factor increases, more pixels need to be reconstructed, and when restoring the HR image, more attention should be given to high-frequency detail features. Otherwise, the reconstructed image may appear blurry. Additionally, in extracting these features, it is not only related to the scale factor, but also related to the local features. For example, the convolutional weights of the edge region are obviously different from those of the smooth region, which provides the necessity for introducing local features into the feature extraction network. Therefore, to activate more information, it is necessary to introduce scale factors and local feature information during the feature extraction process to extract scale-aware local structure adaptive features.

Inspired by the dynamic convolution proposed by Chen *et al.* [18], we propose combining scale and feature information to generate learnable multiple attention weights for multiple parallel convolutional kernels. Then the attention weights are used to aggregate the multiple parallel convolutional kernels into a single kernel. This allows for the extraction of adaptive features based on the input features and scale. The proposed SLFAM is illustrated in Fig. 2.

The dynamic attention weights are generated using a lightweight attention network, inspired by the network structure of SENet [41]. Given the input feature F_{LR} and the two scale factors r_h and r_w, for the vertical and horizontal directions respectively, the vector $[r_h, r_w]$ is first expanded to match the size of F_{LR} in the channel dimension. Then, the expanded vector is concatenated with F_{LR}. The concatenated features go through adaptive average pooling, followed by two fully connected (FC) layers. Finally, the $Softmax$ function is applied to obtain K dynamic attention weights.

The dynamic convolution part consists of K parallel convolutions, where K convolutional kernels of the same size are

aggregated into one kernel using the dynamic attention weights π_k. The process can be expressed as follows:

$$W(F, r_h, r_w) = \sum_{k=1}^{K} \pi_k(F, r_h, r_w) W_k \tag{5}$$

$$b(F, r_h, r_w) = \sum_{k=1}^{K} \pi_k(F, r_h, r_w) b_k \tag{6}$$

Here, W_k and b_k represent the weights and biases of the *K-th* convolutional kernel, respectively. π_k is the attention weight corresponding to the convolutional kernel. The values of π_k satisfy the following conditions:

$$0 \leq \pi_k(F, r_h, r_w) \leq 1 \tag{7}$$

$$\sum_{k=1}^{K} \pi_k(F, r_h, r_w) = 1 \tag{8}$$

To further reduce the number of parameters, inspired by depthwise separable convolution [47], the aggregated single convolutional kernel is split into depthwise convolutional kernels (dw) and pointwise convolutional kernels (pw). The depthwise convolution operates only on the 2D plane without changing the number of channels. Since the depthwise convolution does not utilize the feature information from different channels at the same spatial position, the pointwise convolution is used to combine the features from different channels. These two operations are performed by different dynamic convolution modules:

$$pw = f_{d1}(Concat(F_{LR}, r_h, r_w)) \tag{9}$$

$$dw = f_{d2}(Concat(F_{LR}, r_h, r_w)) \tag{10}$$

Here, F_{LR} is the input feature, r_h and r_w are the scale factors in the horizontal and vertical directions. $f_{d1}(\cdot)$ and $f_{d2}(\cdot)$ represents the dynamic convolution process.

Assuming that the original feature $F \in R^{64\times H\times W}$ is convolved with the original convolutional kernel $W \in R^{64\times 64\times 3\times 3}$, the number of parameters in the kernel is 36864. By using depthwise separable convolution, the number of parameters becomes the sum of the depthwise convolutional kernel $W \in R^{64\times 1\times 3\times 3}$ and the pointwise convolutional kernel $W \in R^{64\times 64\times 1\times 1}$, which is 4672, accounting for only 12.67% of the original convolutional kernel. Moreover, generating pw and dw separately through attention networks allows them to better adapt to the scale and local features at the channel and pixel levels.

The input feature F_{LR} is first convolved with the dw kernel and then convolved with the pw kernel. Finally, the output feature of this module, denoted as F'_{LR}, is obtained:

$$F'_{LR} = F_{LR} * dw * pw \tag{11}$$

Considering the potential instability introduced by dynamic convolution, an annealing strategy [18] is employed. At the beginning of training, a large temperature coefficient τ is added to the $Softmax$ function, which makes the attention weights approach a uniform distribution. The specific process is defined by the following equation:

$$\pi_k = \frac{exp(\zeta_k/\tau)}{\sum_{i=1}^{K} exp(\zeta_i/\tau)} \tag{12}$$

Here, ζ_k represents the input feature of the $Softmax$ function. In the original $Softmax$ function, τ= 1. As it increases, the $Softmax$ output becomes sparser. As the number of iterations increases, the training gradually stabilizes, and the temperature coefficient τ decreases continuously until it reaches 1. This ensures both training stability and accelerated network convergence.

C. Local Feature Adaptation Upsample

For integer scale image SR tasks, in a commonly used upsampling method PixelShuffle [42], the amount of upsampling kernels is pre-defined for each scale factor. However, in the case of arbitrary-scale SR tasks, the mapping relationship between the pixels of the LR image and the corresponding pixels of the HR image is related to scale factors. This means that different scale factors correspond to different mapping functions. Therefore, it is not possible to predefine the number of upsampling filters for arbitrary scale factors.

To address this issue, methods like MetaSR [16] and ArbSR [8] utilize the scale factor as an input to predict the upsampling kernels weights dynamically, enabling the generation of HR images at arbitrary scales. However, these methods only use the scale factor and coordinate information to predict the upsampling kernels weights, without considering that the weights should also be content-dependent.

To achieve this, we propose a Local Feature Adaptation Upsampling Module (LFAUM) that combines the scale factor and the local features of the LR image to jointly predict the upsampling kernels weights. The predicted upsampling kernels determine the value of each SR pixel based on its corresponding nearest neighboring LR pixel. However, since the number of HR pixels is much greater than the number of LR pixels, it is inevitable that multiple HR pixels will map to the same LR pixel. Meanwhile, the impact of local content differences of LR image should also be taken into account in the reconstruction of the upsampling kernel weight of SR pixel. In such cases, it is necessary to introduce scale information, position information and local features to predict the upsampling kernels weights and to distinguish different HR pixels that mapped to the same LR pixel.

The structure of the LFAUM is illustrated in Fig. 3. The input LR feature $F_{LR} \in R^{C_{in}\times H\times W}$ is fed into this upsampling module to obtain the output HR feature $F_{HR} \in R^{C_{out}\times(r_h H)\times(r_w W)}$, where C_{in} and C_{out} represent the input and output channels, respectively. Finally, the feature F_{HR} is mapped onto the HR image space using convolutions.

From Fig. 3, it can be seen that the relative position matrix P consists of four parts, including the scale factors r_h and r_w for the image's horizontal and vertical dimensions, as well as the relative position matrices $R(x)$ and $R(y)$. The purpose of taking the reciprocal of the scale factors r_h and r_w is to map the data onto the range of 0 to 1, consistent with the size of

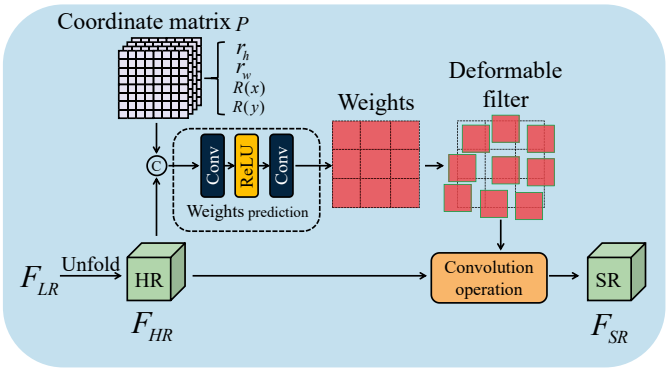


Fig. 3. The structure of local feature adaptation upsampling module.

the input image data. The relative position matrix P contains the coordinate offset relationships when projecting HR pixel values to the LR image space. For each pixel (x, y) in the HR space, the corresponding LR pixel position coordinates $L(x)$ and $L(y)$ after direct projection to the LR space are calculated as follows:

$$L(x) = \frac{x + 0.5}{r_h} - 0.5 \tag{13}$$

$$L(y) = \frac{y + 0.5}{r_w} - 0.5 \tag{14}$$

Since the obtained x and y coordinates from the above equations will be projected onto the continuous space of the LR image size, while the actual LR image coordinates are discrete, we introduce the relative position coordinates $R(x)$ and $R(y)$:

$$R(x) = L(x) - \lfloor \frac{x + 0.5}{r_h} \rfloor \tag{15}$$

$$R(y) = L(y) - \lfloor \frac{y + 0.5}{r_w} \rfloor \tag{16}$$

The relative position coordinates $R(x)$ and $R(y)$ are obtained by calculating the relative positional offsets of $L(x)$ and $L(y)$ to the nearest LR image coordinates. Their values are constrained within the range of 0 to 1.

To facilitate the convolution operation, the F_{LR} is unfolded using the unfold function according to the $k \times k$ kernel, resulting in a feature matrix of size $F'_{LR} \in R^{C_{in} \times (k*k) \times H \times W}$. Then, based on the mapping function from HR pixels to the corresponding LR pixel values, the unfolded features are transformed into $F_{HR} \in R^{C_{out} \times (r_h H) \times (r_w W)}$. F_{HR} is then concatenated with the relative position matrix P, and the filter weights are generated using the following process:

$$W = f(Concat(F_{HR}, P)) \tag{17}$$

where $f(\cdot)$ consists of two 1×1 convolutions and an activation function.

However, due to the differences in upsampling scale factors and position, fixed-shaped upsampling filters may cause certain deformations in objects. Therefore, for visual tasks that require fine localization, such as SR, it is necessary to determine the size and range of the receptive field adaptively.

In order to activate more information for superior upsampling, we learn additional offset values of the receptive field from the LR features. These offset values are then used to adaptively change the spatial sampling positions during the reconstruction process, thereby enhancing the focus on pertinent image regions. Additionally, for each pixel's upsampling filter, we not only learn the offset values of the receptive field but also learn the feature modulation amplitude. This allows the module to alter the spatial distribution and relative influence of the features, further enhancing the geometric transformation modeling ability of the entire network.

The specific process is as follows:

$$R = (-1, -1), (-1, 0), ..., (1, 1) \tag{18}$$

The grid R defines the size and expansion of the receptive field, which corresponds to a convolutional kernel with a size of 3×3. For each position p_0 on the output feature map F_{SR}, we have:

$$F_{SR}(p_0) = \sum_{p_n \in R} w(p_n) * F_{HR}(p_0 + p_n) \tag{19}$$

where p_n enumerates the positions in R.

In our deformable upsampling, the regular grid R is adaptively adjusted based on the learned offsets $\delta p_n | n = 1, 2, ..., N$, and the feature distribution is modified based on the learned modulation scalar Δm_p. Here, N is the number of points in the grid R, δp_n includes the offsets in both the horizontal and vertical directions, and Δm_p ranges from 0 to 1. The equation is modified as follows:

$$F_{SR}(p_0) = \sum_{p_n \in R} w(p_n) * F_{HR}(p_0 + p_n + \delta p_n) * \Delta m_p \tag{20}$$

Now, the upsampling is performed at irregular offset positions $(p_0 + p_n + \delta p_n)$. Since δp_n is usually a fractional value, it cannot directly correspond to the feature value of a pixel. Therefore, the feature value at the sampling point is obtained through bilinear interpolation with the nearest four neighboring pixels.

The final feature values of the output image F_{SR} are determined by the combination of F_{HR} and the deformed upsampling dynamic convolutional kernel W_{dconv}:

$$F_{SR}(x, y) = F_{HR}(x, y) * W_{dconv} \tag{21}$$

The introduction of deformable convolution makes the network have stronger geometric modeling ability during upsampling, so as to adapt to the characteristics of the image content itself and avoid the offset caused by the fixed shape upsampling kernels.

The incorporation of scale information, relative positions, local features, and receptive field offsets in the deformed upsampling dynamic convolutional kernel allows for a more detailed reconstruction of the image, approaching the quality of the HR image.

IV. Experiments

In this section, we first introduce the datasets, metrics and implementation details. Then, we compare our model with the SOTA methods on both symmetric and asymmetric scale factors.

TABLE I
AVERAGE RESULTS OF PSNR (dB) AND SSIM FOR SYMMETRIC SCALE FACTORS ON FIVE BENCHMARK DATASETS. THE BEST AND SECOND-BEST RESULTS ARE MARKED IN RED AND BLUE COLORS, RESPECTIVELY.

Methods	Metrics	Set5			Set14			BSD100			Urban100			Manga109		
		×2	×1.6	×1.55	×2	×1.5	×1.65	×2	×1.4	×1.85	×2	×1.9	×1.95	×2	×1.7	×1.95
Bicubic	PSNR	33.66	36.10	36.24	30.24	32.87	31.83	29.56	32.95	30.11	26.88	27.25	27.05	30.80	32.91	31.12
	SSIM	0.9299	0.9555	0.9578	0.8688	0.9268	0.9020	0.8431	0.9031	0.8581	0.8410	0.8510	0.8460	0.9339	0.9579	0.9379
RCAN [38]	PSNR	38.27	40.53	40.77	34.12	37.23	36.08	32.40	36.86	33.16	33.18	33.17	32.84	39.42	41.15	39.39
	SSIM	0.9614	0.9755	0.9749	0.9216	0.9580	0.9482	0.9021	0.9639	0.9202	0.9374	0.9441	0.9401	0.9788	0.9897	0.9804
MetaSR [16]	PSNR	38.22	40.63	40.93	34.04	37.51	36.17	32.35	36.88	33.22	33.12	33.62	33.30	39.32	41.30	39.59
	SSIM	0.9612	0.9754	0.9757	0.9213	0.9575	0.9480	0.9019	0.9639	0.9194	0.9358	0.9425	0.9397	0.9782	0.9891	0.9803
ArbSR [8]	PSNR	38.24	40.66	40.97	34.09	37.53	36.28	32.39	36.87	33.23	33.14	33.55	33.25	39.37	41.32	39.56
	SSIM	0.9613	0.9754	0.9759	0.9215	0.9581	0.9476	0.9021	0.9644	0.9206	0.9367	0.9439	0.9400	0.9785	0.9896	0.9803
LIIF [45]	PSNR	38.18	40.61	41.00	33.97	37.45	36.25	32.32	36.91	33.14	32.87	33.52	33.20	39.21	41.32	39.52
	SSIM	0.9610	0.9753	0.9751	0.9210	0.9578	0.9472	0.9011	0.9632	0.9191	0.9352	0.9425	0.9386	0.9774	0.9886	0.9794
LTE [46]	PSNR	38.30	40.69	41.18	34.25	37.69	36.45	32.44	36.92	33.26	33.38	34.11	33.83	39.53	41.69	39.87
	SSIM	0.9615	0.9756	0.9755	0.9231	0.9599	0.9498	0.9023	0.9646	0.9198	0.9388	0.9454	0.9427	0.9791	0.9903	0.9810
CLIT [56]	PSNR	38.26	40.69	41.11	34.21	37.68	36.38	32.39	36.94	33.25	33.13	34.09	33.85	39.49	41.62	39.81
	SSIM	0.9614	0.9751	0.9754	0.9224	0.9588	0.9489	0.9020	0.9639	0.9199	0.9363	0.9437	0.9430	0.9789	0.9897	0.9809
Ours	PSNR	38.34	40.73	41.15	34.48	37.79	36.54	32.48	36.98	33.35	33.51	34.35	34.03	39.65	41.67	39.95
	SSIM	0.9616	0.9756	0.9760	0.9245	0.9610	0.9510	0.9026	0.9645	0.9205	0.9406	0.9475	0.9442	0.9793	0.9902	0.9812
		×3	×2.4	×2.75	×3	×2.8	×2.95	×3	×2.2	×2.15	×3	×2.3	×2.35	×3	×2.7	×2.55
Bicubic	PSNR	30.39	32.41	31.06	27.55	27.84	27.46	27.21	28.88	29.12	24.46	25.91	25.72	26.95	27.77	28.27
	SSIM	0.8682	0.9072	0.8845	0.7742	0.7932	0.7790	0.7385	0.8185	0.8235	0.7370	0.8070	0.8020	0.8556	0.8790	0.8907
RCAN [38]	PSNR	34.76	36.51	35.31	30.62	30.90	30.53	29.31	31.31	31.68	29.01	31.34	31.15	34.42	35.50	36.06
	SSIM	0.9299	0.9484	0.9364	0.8482	0.8630	0.8538	0.8111	0.8811	0.8873	0.8702	0.9181	0.9155	0.9499	0.9607	0.9654
MetaSR [16]	PSNR	34.76	36.58	35.36	30.58	31.00	30.56	29.29	31.44	31.70	28.96	31.43	31.20	34.40	35.55	36.21
	SSIM	0.9305	0.9482	0.9377	0.8471	0.8616	0.8528	0.8099	0.8801	0.8863	0.8665	0.9144	0.9118	0.9488	0.9591	0.9643
ArbSR [8]	PSNR	34.76	36.59	35.39	30.64	31.01	30.59	29.32	31.48	31.72	28.98	31.48	31.26	34.55	35.64	36.27
	SSIM	0.9306	0.9487	0.9375	0.8477	0.8622	0.8534	0.8115	0.8815	0.8877	0.8674	0.9156	0.9127	0.9501	0.9608	0.9656
LIIF [45]	PSNR	34.68	36.47	35.38	30.53	30.97	30.56	29.26	31.47	31.67	28.82	31.30	31.09	34.17	35.49	36.18
	SSIM	0.9293	0.9479	0.9356	0.8470	0.8615	0.8527	0.8099	0.8801	0.8858	0.8663	0.9142	0.9116	0.9478	0.9585	0.9633
LTE [46]	PSNR	34.89	36.66	35.51	30.80	31.28	30.83	29.39	31.59	31.78	29.29	31.95	31.77	34.77	35.94	36.52
	SSIM	0.9318	0.9504	0.9381	0.8502	0.8647	0.8559	0.8128	0.8831	0.8888	0.8730	0.9218	0.9173	0.9508	0.9616	0.9663
CLIT [56]	PSNR	34.80	36.69	35.47	30.78	31.30	30.72	29.34	36.93	31.75	29.04	31.85	31.79	34.63	35.80	36.39
	SSIM	0.9310	0.9496	0.9373	0.8503	0.8652	0.8560	0.8124	0.8824	0.8884	0.8706	0.9185	0.9159	0.9503	0.9610	0.9659
Ours	PSNR	34.97	36.73	35.54	30.90	31.36	30.87	29.45	31.58	31.83	29.37	32.06	31.86	34.86	36.01	36.57
	SSIM	0.9316	0.9499	0.9382	0.8510	0.8657	0.8565	0.8137	0.8838	0.8898	0.8741	0.9223	0.9191	0.9513	0.9617	0.9670
		×4	×3.1	×3.25	×4	×3.2	×3.95	×4	×3.2	×3.55	×4	×3.7	×3.85	×4	×3.4	×3.65
Bicubic	PSNR	28.42	29.89	29.21	26.00	26.98	26.32	25.96	26.91	26.32	23.14	23.38	23.14	24.89	25.97	25.41
	SSIM	0.8104	0.8622	0.8532	0.7227	0.7595	0.7252	0.6675	0.7235	0.6990	0.6585	0.6819	0.6702	0.7866	0.8280	0.8108
RCAN [38]	PSNR	32.63	34.37	33.92	28.85	30.00	28.72	27.75	28.86	28.27	26.75	27.20	26.89	31.20	32.76	32.04
	SSIM	0.9002	0.9264	0.9232	0.7889	0.8336	0.7929	0.7436	0.7953	0.7780	0.8087	0.8285	0.8182	0.9173	0.9585	0.9417
MetaSR [16]	PSNR	32.56	34.46	33.98	28.85	30.08	28.73	27.75	28.86	28.30	26.71	27.25	26.93	31.33	33.00	32.22
	SSIM	0.8984	0.9245	0.9216	0.7872	0.8329	0.7902	0.7423	0.7932	0.7774	0.8083	0.8276	0.8183	0.9180	0.9592	0.9424
ArbSR [8]	PSNR	32.55	34.50	34.03	28.87	30.08	28.74	27.76	28.93	28.33	26.68	27.22	26.90	31.36	33.12	32.29
	SSIM	0.8992	0.9251	0.9225	0.7883	0.8330	0.7923	0.7441	0.7958	0.7785	0.8047	0.8245	0.8142	0.9196	0.9611	0.9436
LIIF [45]	PSNR	32.50	34.47	34.12	28.80	30.09	28.85	27.74	28.90	28.34	26.68	27.23	26.94	31.20	32.87	32.11
	SSIM	0.8988	0.9256	0.9212	0.7874	0.8321	0.7914	0.7420	0.7937	0.7764	0.8040	0.8238	0.8135	0.9170	0.9582	0.9414
LTE [46]	PSNR	32.81	34.69	34.16	28.94	30.33	28.87	27.81	29.00	28.42	26.94	27.86	27.40	31.67	33.42	32.61
	SSIM	0.9018	0.9286	0.9242	0.7904	0.8351	0.7944	0.7450	0.7969	0.7799	0.8100	0.8298	0.8195	0.9115	0.9527	0.9359
CLIT [56]	PSNR	32.69	34.60	34.17	28.98	30.22	28.79	27.82	28.95	28.40	26.91	27.81	27.31	31.58	33.24	32.56
	SSIM	0.9005	0.9263	0.9238	0.7911	0.8367	0.7945	0.7452	0.7967	0.7794	0.8094	0.8285	0.8193	0.9203	0.9624	0.9446
Ours	PSNR	32.83	34.69	34.20	29.00	30.31	28.89	27.88	29.04	28.45	27.03	27.94	27.46	31.69	33.50	32.69
	SSIM	0.9017	0.9281	0.9245	0.7918	0.8368	0.7955	0.7462	0.7976	0.7809	0.8130	0.8325	0.8228	0.9205	0.9619	0.9447

A. Datasets and Metrics

We use the DIV2K dataset [48] as the training set and evaluate the model on five benchmark datasets, including Set5 [49], Set14 [50], BSD100 [51], Urban100 [52], and Manga109 [53]. Following the settings of previous works, we use the peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) [54] calculated on the transformed Y channel in the YCbCr space as the evaluation metrics. We crop borders for fair comparison.

B. Implementation Details

Following ArbSR [8], for symmetric scale factors, where the vertical and horizontal reconstruction factors are the same, we generate LR training images for each scale factor from 1 to 4 with a stride of 0.1 (*i.e.*, 1.1, 1.2, ..., 3.9, 4.0) during the training phase. For asymmetric scale factors, we generate LR training images for each asymmetric scale factor by varying the vertical and horizontal factors with a stride of 0.5 (*i.e.*, $\frac{\times 1.5}{\times 2.0}$, $\frac{\times 1.5}{\times 2.5}$, ..., $\frac{\times 4.0}{\times 3.0}$, $\frac{\times 4.0}{\times 3.5}$). In each batch, we randomly

TABLE II
AVERAGE RESULTS OF PSNR (dB) AND SSIM FOR ASYMMETRIC SCALE FACTORS ON FIVE BENCHMARK DATASETS. THE BEST AND SECOND-BEST RESULTS ARE MARKED IN RED AND BLUE COLORS, RESPECTIVELY.

Methods	Metrics	Set5			Set14			BSD100			Urban100			Manga109		
		$\frac{\times 1.5}{\times 4}$	$\frac{\times 1.5}{\times 3.5}$	$\frac{\times 1.6}{\times 3.05}$	$\frac{\times 4}{\times 2}$	$\frac{\times 3.5}{\times 2}$	$\frac{\times 3.5}{\times 1.75}$	$\frac{\times 4}{\times 1.4}$	$\frac{\times 1.5}{\times 3}$	$\frac{\times 3.5}{\times 1.45}$	$\frac{\times 1.6}{\times 3}$	$\frac{\times 1.6}{\times 3.8}$	$\frac{\times 3.35}{\times 1.55}$	$\frac{\times 2.5}{\times 2}$	$\frac{\times 2.8}{\times 3.5}$	$\frac{\times 3.35}{\times 2.7}$
Bicubic	PSNR	30.01	30.83	31.40	27.25	27.88	27.27	27.45	28.86	27.94	25.93	24.92	25.19	29.61	26.47	26.86
	SSIM	0.8640	0.8792	0.8877	0.7663	0.7937	0.7668	0.7501	0.8180	0.7739	0.8076	0.7541	0.7714	0.9135	0.8418	0.8537
RCAN [38]	PSNR	34.14	35.05	35.67	30.35	31.02	d31.21	29.35	31.30	29.98	30.72	28.81	29.34	37.48	33.31	33.82
	SSIM	0.9230	0.9359	0.9397	0.8419	0.8638	0.8726	0.8125	0.8776	0.8394	0.9069	0.8687	0.8963	0.9715	0.9493	0.9529
MetaSR [16]	PSNR	34.20	35.17	35.81	30.40	31.05	31.33	29.43	31.26	30.09	30.73	29.03	29.67	37.74	33.61	34.23
	SSIM	0.9235	0.9364	0.9403	0.8435	0.8655	0.8742	0.8135	0.8786	0.8404	0.9070	0.8688	0.8964	0.9727	0.9515	0.9531
ArbSR [8]	PSNR	34.37	35.40	36.05	30.55	31.27	31.54	29.54	31.40	30.22	31.13	29.36	30.04	37.93	33.81	34.41
	SSIM	0.9246	0.9375	0.9414	0.8467	0.8687	0.8774	0.8147	0.8798	0.8416	0.9116	0.8734	0.9010	0.9736	0.9524	0.9541
LIIF [45]	PSNR	34.30	35.29	35.99	30.51	31.16	31.56	29.47	31.35	30.22	30.95	29.28	29.92	37.88	33.74	34.34
	SSIM	0.9244	0.9373	0.9412	0.8462	0.8673	0.8777	0.8140	0.8791	0.8409	0.9096	0.8714	0.8990	0.9734	0.9512	0.9546
LTE [46]	PSNR	34.53	35.48	36.14	30.70	31.52	31.69	29.62	31.45	30.27	31.33	29.61	30.28	38.02	34.01	34.69
	SSIM	0.9260	0.9389	0.9428	0.8492	0.8703	0.8807	0.8155	0.8806	0.8424	0.9140	0.8758	0.9034	0.9740	0.9527	0.9545
CLIT [56]	PSNR	34.47	35.49	36.11	30.67	31.44	31.70	29.58	31.42	30.25	31.26	29.53	30.32	37.99	33.95	34.58
	SSIM	0.9252	0.9381	0.9420	0.8481	0.8701	0.8808	0.8151	0.8802	0.8420	0.9131	0.8749	0.9035	0.9739	0.9527	0.9543
Ours	PSNR	34.62	35.61	36.20	30.83	31.60	31.85	29.67	31.51	30.33	31.49	29.70	30.41	38.27	34.12	34.85
	SSIM	0.9269	0.9397	0.9438	0.8506	0.8723	0.8816	0.8158	0.8805	0.8429	0.9158	0.8772	0.9056	0.9753	0.9537	0.9561

crop patches of size 50×50 as inputs. We also apply random vertical or horizontal flipping and randomly rotate the images by 90°for data augmentation. The Adam method [55] is used as optimization with β_1=0.9 and β_2=0.99. The initial learning rate is set to 1×10^{-4} and is halved every 30 epochs. We train our model for 150 epochs with a batch size of 8. The loss function used is L_1 loss. To address the issue of gradient explosion when training arbitrary-scale SR models, we use the pretrained model of the RCAN [38] network with ×4 scale factor. Additionally, only integer scale factors (*i.e.*, ×2, ×3, ×4) are used for training in the first epoch. Non-integer scale factors are randomly selected starting from the second epoch to ensure training stability. Our model is trained on two Nvidia GeForce GTX 1080Ti GPUs using the PyTorch framework.

C. Results for SR with Symmetric Scale Factors

In this section, we compare our proposed network with RCAN [38], MetaSR [16], ArbSR [8], LIIF [45], LTE [46], and CLIT [56] on SR tasks with symmetric scale factors (both integer and non-integer scale factors). MetaSR [16] introduces a groundbreaking meta-upsampling module for arbitrary-scale upsampling. ArbSR [8] is the first method to achieve asymmetric scale factors reconstruction in arbitrary-scale SR. LIIF [45] and LTE [46] are the SOTA methods in arbitrary-scale SR based on implicit function. The CLIT [56], based on Transformer, is also a SOTA arbitrary-scale SR method. Since the RCAN [38] network itself only supports integer scale factors of ×2, ×3, and ×4, we handle non-integer scale factors using RCAN+Bicubic method. For example, for a scale factor of ×2.7, we first perform ×3 SR using the RCAN [38] network on the LR image, and then downsample the reconstructed image using bicubic interpolation to obtain the desired ×2.7 SR outcome. To ensure a fair comparison, we retrain the models using the official released code.

1) Quantitative Results: Table I provides the average PSNR and SSIM of different algorithms on the Set5, Set14, BSD100, Urban100, and Manga109 benchmark datasets. From the table, it can be observed that our network shows improvements over the baseline RCAN [38] for integer scale factors, and significantly outperforms RCAN [38] for non-integer scale factors. For example, our network achieves better performance than RCAN [38] for ×2 SR on Set5 (PSNR: 38.34 dB vs. 38.27 dB; SSIM: 0.9616 vs. 0.9614), and shows a more significant advantage for ×1.55 SR (PSNR: 41.15 dB vs. 40.77 dB; SSIM: 0.9760 vs. 0.9749).

Compared to SOTA methods, our network is better than them overall with most scale factors. For example, compared to LTE [46], the PSNR of our network improves by 0.24 dB on ×1.9 scale factors SR on Urban100 while the SSIM improves by 0.0021.

It is worth noting that our method performs better on the Urban100 dataset compared to the other benchmark datasets. This is mainly due to the fact that the Urban100 dataset consists of various architectural images with richer object edges and details. Our method, which considers scale, position, and local feature information to generate upsampling filters, achieves better restoration results in edge details compared to the other methods.

2) Qualitative Results: Fig. 4 presents the visual results of different SR algorithms at multiple symmetric scales. In particular, Fig. 4(a) shows the SR result of the image "img013" from the Urban100 dataset at a symmetric scale factor of ×1.9. From the results, it is evident that most of the comparative algorithms produce blurred outputs and fail to reconstruct parallel lines clearly. In contrast, our algorithm can perceive dense lines and generate finer details. The reconstructed result in Fig. 4(c) demonstrates that only our model can faithfully reconstruct the triangular patterns in the image, yielding better perceptual quality and fewer artifacts. In terms of visual

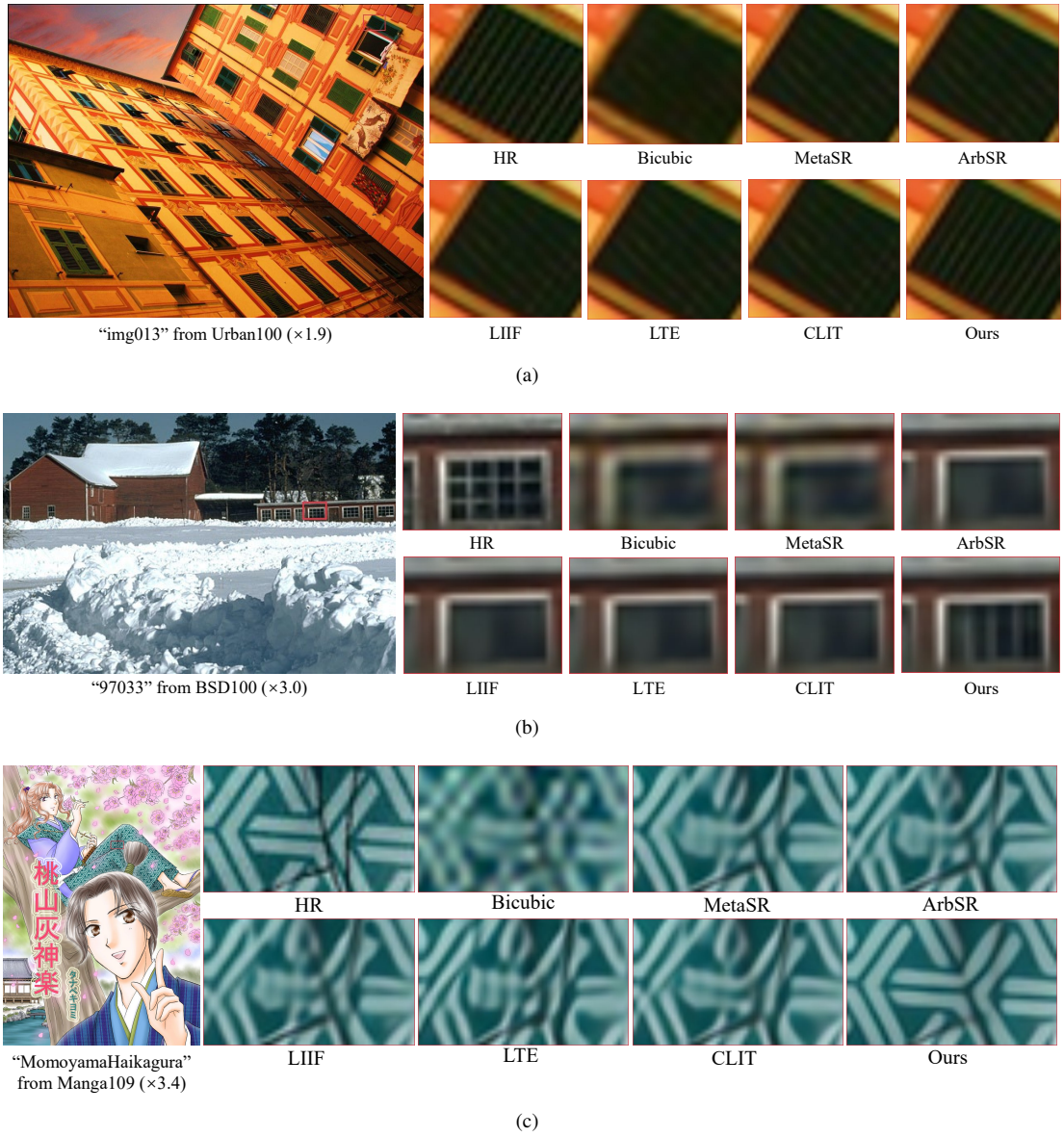


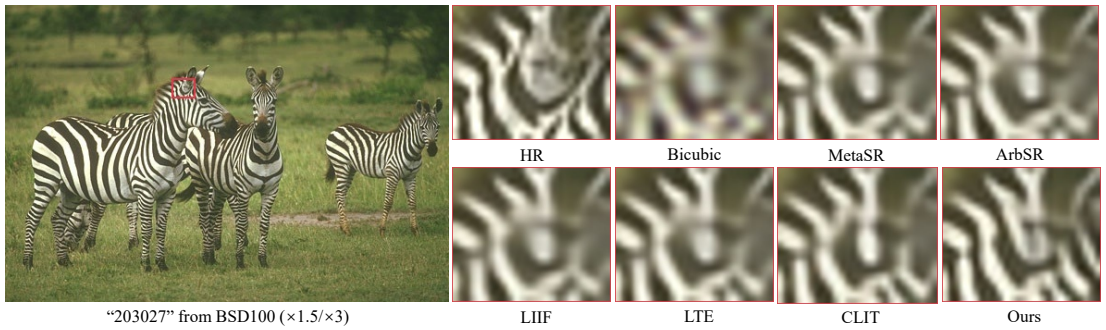
Fig. 4. Visual comparison for SR with symmetric scale factors.

effects, we observe that our model exhibits significant advantages in recovering intersecting grids, repetitive textures, and other high-frequency dense details, which are sensitive to scale changes. This is attributed to the incorporation of scale and local feature information during the upsampling process in our model, enabling the upsampling convolutional kernels to undergo geometric transformations based on local features. As a result, the reconstructed outputs adapt well to the image content.

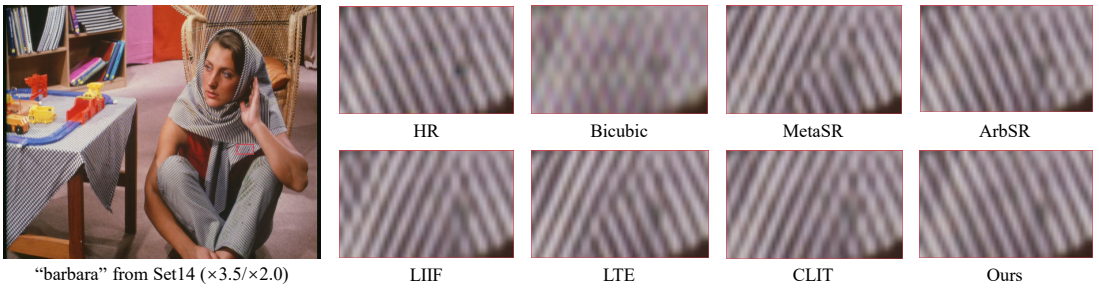
D. Results for SR with Asymmetric Scale Factors

In this section, we conduct a comparison between our proposed model and several existing methods including RCAN [38], MetaSR [16], ArbSR [8], LIIF [45], LTE [46], and CLIT [56]. Except for ArbSR [8], the other algorithms do not support asymmetric scale factors in both horizontal and vertical directions. To facilitate the comparison, we adopt the RCAN+Bicubic approach. For example, when the scale factor is $\times 1.5/\times 3.0$, we first reconstruct the LR image using the comparison algorithm at a scale factor of $\times 3$, and then downsample the reconstructed image using bicubic interpolation to achieve the desired size for the horizontal scale factor of $\times 1.5$. This process yields the $\times 1.5/\times 3.0$ scale factor image. We evaluate their performance on SR tasks with asymmetric scale factors.

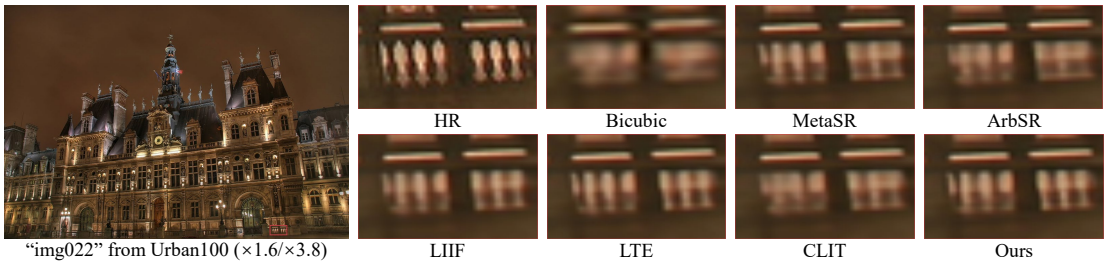
1) Quantitative Results: Table II presents the average PSNR and SSIM of different algorithms on the Set5, Set14, BSD100, Urban100 and Manga109 benchmarks. Since algorithms like MetaSR [16] and LIIF [45] do not support asymmetric scales, their performance is affected by the downsampling operation, resulting in less improvement compared to the RCAN [38] network. Our model takes into account the different effects of horizontal and vertical scaling on image SR during training and incorporates LR images with asymmetric scales. Therefore, our model exhibits significant performance improvement. For instance, our method achieves noticeable



(a)



(b)



(c)

Fig. 5. Visual comparison for SR with asymmetric scale factors.

improvement over SOTA algorithms such as LTE [46] and CLIT [56] on the $\times 2.5/\times 2.0$ SR task in Manga109 dataset (PSNR: 38.27 dB vs. 38.02 dB vs. 37.99 dB; SSIM: 0.9753 vs. 0.9740 vs. 0.9739). Clearly, this is attributed to the diverse information utilized by our model, including local features and scales, allowing for better detail restoration based on the content of the image.

2) Qualitative Results: Fig. 5 showcases the visual results of different SR reconstruction algorithms on multiple asymmetric scale factors. In Fig. 5(b), it can be observed that most SR results suffer from blurred textures, incorrect patterns, and significant loss of edge details. In contrast, our model excels in generating fine texture details while accurately perceiving local line orientations, leading to superior visual quality. In Fig. 5(a), it is evident that most algorithms fail to faithfully restore repetitive textures, resulting in distortion artifacts, whereas only our method successfully recovers dense lines with clarity.

Our model effectively utilizes the features within the neighborhood, enhancing the focus on pertinent image regions and demonstrating better perceptual capabilities for edge and high-frequency details. This ability minimizes deviations during the reconstruction process and preserves the original dense repetitive structures, contributing to the outstanding visual performance of our model on asymmetric scale factor SR tasks.

E. Generalization Performance

In this section, we compare our method with MetaSR [16], ArbSR [8], LIIF [45], LTE [46], and CLIT [56] on generalization performance. We first compare the performance of these methods on out-of-scale ($>\times 4$) SR experiments. Then, we provide the visualization results of our proposed method on real-world images.

1) Results for Out-of-scale SR: To explore the generalization performance on unseen scales, we evaluate the PSNR and SSIM of different methods for out-of-scale SR on five benchmark datasets and the DIV2K validation set [48], as shown in Table III and Table IV. The reason for including the DIV2K validation set here is that its image resolution is large, which is suitable for large-scale SR task. Although the scale factors in the training distribution is $\times 1$ to $\times 4$ only, our proposed method still shows strong representation capability on large scale factors since the method fully considers the

TABLE III
AVERAGE RESULTS OF PSNR (dB) AND SSIM FOR OUT-OF-SCALE SR ON FIVE BENCHMARK DATASETS. THE BEST AND SECOND-BEST RESULTS ARE MARKED IN RED AND BLUE COLORS, RESPECTIVELY.

Methods	Metrics	Set5			Set14			BSD100			Urban100			Manga109		
		×6	×6.25	×8	×5.25	$\frac{\times 5.8}{\times 6.2}$	×8	×6	×6.75	$\frac{\times 7.5}{\times 8}$	×5.75	×6	$\frac{\times 6.25}{\times 6.75}$	$\frac{\times 6.3}{\times 5.6}$	×6.75	×8
Bicubic	PSNR	24.17	23.70	22.74	23.23	23.10	21.12	23.69	23.11	22.79	20.73	20.82	20.53	21.51	20.95	19.81
	SSIM	0.6732	0.6502	0.5992	0.6118	0.6052	0.4966	0.5728	0.5333	0.5107	0.5796	0.5850	0.5673	0.6557	0.6353	0.5973
MetaSR [16]	PSNR	29.04	28.55	26.96	27.10	26.47	24.97	25.90	25.24	24.87	24.37	24.04	23.57	26.99	25.99	24.73
	SSIM	0.8432	0.8309	0.7842	0.7791	0.7557	0.6941	0.7052	0.6687	0.6471	0.7655	0.7516	0.7307	0.8407	0.8161	0.7763
ArbSR [8]	PSNR	29.06	28.60	27.05	27.17	26.47	25.01	25.89	25.30	24.85	24.40	24.07	23.61	27.10	26.08	24.90
	SSIM	0.8437	0.8322	0.7872	0.7816	0.7557	0.6958	0.7047	0.6721	0.6459	0.7667	0.7529	0.7325	0.8429	0.8186	0.7821
LIIF [45]	PSNR	29.15	28.73	27.14	27.38	26.60	25.15	25.98	25.53	24.95	24.44	24.20	23.75	27.30	26.37	25.04
	SSIM	0.8458	0.8355	0.7901	0.7890	0.7607	0.7019	0.7095	0.6850	0.6518	0.7684	0.7584	0.7388	0.8467	0.8262	0.7868
LTE [46]	PSNR	29.32	29.03	27.26	27.46	26.68	25.16	26.01	25.61	25.03	24.79	24.62	23.93	27.78	26.41	25.12
	SSIM	0.8498	0.8429	0.7939	0.7917	0.7637	0.7023	0.7110	0.6895	0.6565	0.7824	0.7757	0.7468	0.8544	0.8273	0.7895
CLIT [56]	PSNR	29.41	28.97	27.34	27.49	26.66	25.35	26.08	25.63	25.01	24.86	24.66	23.91	27.90	26.69	25.30
	SSIM	0.8519	0.8415	0.7964	0.7928	0.7629	0.7105	0.7147	0.6906	0.6553	0.7851	0.7773	0.7459	0.8560	0.8341	0.7953
Ours	PSNR	29.45	29.09	27.40	27.58	26.79	25.42	26.07	25.66	25.05	24.98	24.73	24.04	27.88	26.79	25.41
	SSIM	0.8528	0.8444	0.7982	0.7959	0.7678	0.7135	0.7142	0.6922	0.6577	0.7897	0.7800	0.7516	0.8558	0.8363	0.7988

TABLE IV
AVERAGE RESULTS OF PSNR (dB) AND SSIM ON DIV2K VALIDATION SET. THE BEST AND SECOND-BEST RESULTS ARE MARKED IN RED AND BLUE COLORS, RESPECTIVELY.

Methods	Metrics	In-scale			Out-of-scale						
		×2	×3	×4	×6	×8.5	×12	×15.5	×18	×24	×30
Bicubic	PSNR	31.01	28.22	26.66	24.82	23.45	22.27	21.42	21.00	20.19	19.59
	SSIM	0.8942	0.8128	0.7602	0.6963	0.6500	0.6128	0.5881	0.5768	0.5567	0.5436
MetaSR [16]	PSNR	35.00	31.27	29.25	26.88	24.83	23.73	22.69	22.18	21.17	20.47
	SSIM	0.9541	0.9005	0.8454	0.7678	0.6966	0.6593	0.6257	0.6101	0.5813	0.5634
ArbSR [8]	PSNR	34.96	31.24	29.26	26.90	24.89	23.77	22.75	22.25	21.23	20.50
	SSIM	0.9540	0.8998	0.8457	0.7685	0.6987	0.6606	0.6276	0.6122	0.5829	0.5641
LIIF [45]	PSNR	34.99	31.26	29.27	26.99	25.02	23.89	22.84	22.34	21.31	20.59
	SSIM	0.9541	0.9003	0.8460	0.7715	0.7032	0.6646	0.6304	0.6149	0.5851	0.5663
LTE [46]	PSNR	35.04	31.32	29.33	27.04	25.07	23.95	22.90	22.40	21.36	20.64
	SSIM	0.9542	0.9017	0.8478	0.7733	0.7049	0.6666	0.6323	0.6167	0.5865	0.5675
CLIT [56]	PSNR	35.10	31.38	29.40	27.12	25.11	24.01	22.96	22.45	21.38	20.64
	SSIM	0.9544	0.9031	0.8499	0.7760	0.7063	0.6687	0.6342	0.6183	0.5870	0.5673
Ours	PSNR	35.17	31.46	29.46	27.16	25.17	24.06	22.99	22.49	21.40	20.66
	SSIM	0.9546	0.9049	0.8517	0.7774	0.7084	0.6703	0.6352	0.6195	0.5876	0.5680

impact of scale factors on dynamic attention weights in feature extraction. For example, as shown in Table IV, our method achieves clear improvement over LTE [46] on the ×18 SR task on DIV2K validation set (PSNR: 22.49 dB vs. 22.40 dB; SSIM: 0.6195 vs. 0.6167). Due to the ability to flexibly recover images of the proposed LFAUM, our model also exhibits strong modeling ability on asymmetric large scale factors. For instance, as shown in Table III, on the ×6.25/×6.75 SR task in Urban100, our network outperforms the SOTA method CLIT [56] (PSNR: 24.04 dB vs. 23.91 dB; SSIM: 0.7516 vs. 0.7459). The visual comparisons for SR algorithms on out-of-scale SR task are shown in Fig. 6. From the results, our proposed network is still able to reconstruct SR in clearer and more natural details at both symmetric and asymmetric scales for out-of-scale SR. In Fig. 6(c), it is shown that our method can clearly recover the lines of the window with less blurring and artifacts, which reveals that the model has excellent generalization performance.

2) Visualization Results of Real-world Images: To evaluate the generalization performance of our model for real-world image SR, we test our network on real-world images using RealSR [57] dataset, and present the results in Fig. 7. Compared to the conventional interpolation method, our proposed network has sharper and more natural edges. For example, as shown in Fig. 7(a), the characters in the street sign reconstructed by our method are still recognizable. These results show that the proposed method still maintains a satisfactory capability in the SR task of real-world images.

F. Ablation Study

In this section, firstly, we conduct ablation experiments to validate the effectiveness of each module. The dataset we use is Urban100 [52], if not specified. We use RCAN [38] as the baseline and introduce six variants, all of which are trained

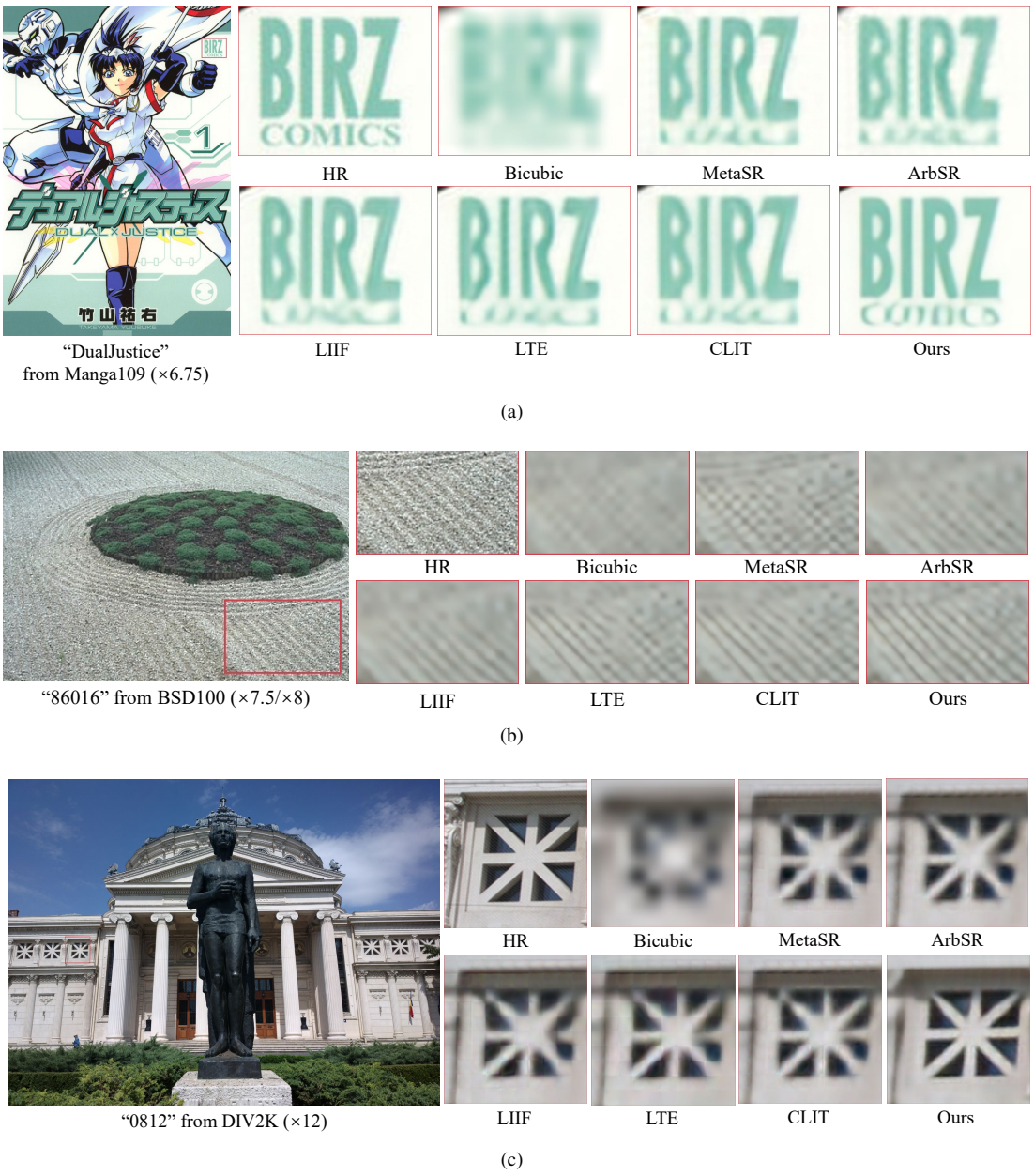


Fig. 6. Visual comparison for SR on out-of-scale factors.

for 150 epochs. Secondly, we discuss the model complexity and inference time.

1) SLFAM and LFAUM Ablation Study: The main innovation of this paper lies in the proposal of two efficient and pluggable modules, namely SLFAM and LFAUM. In the SLFAM, dynamic convolution is utilized to generate dynamic convolutional kernels and incorporate local feature information during the process to enable the learned kernels to adapt to different regions local structures. Subsequently, we employ depth separable convolution to reduce model parameters and enhance the representation capacity of dynamic convolutional kernels. In the LFAUM, we integrate scale and local feature information to generate adaptive upsampling convolutional kernels and use depth separable convolution to achieve adaptive offset of the receptive field of upsampling convolutional kernels, enhancing the network's capability to model geometric transformations. To validate the effectiveness of these components, we conduct ablation experiments on these two modules.

a) For LFAUM: we use RCAN [38] as the baseline. The bicubic method mentioned earlier is adopted for downsampling for non-integer or non-symmetric scale factor SR. To enable RCAN [38] to achieve arbitrary-scale SR within a single model, a simple solution is replacing the PixelShuffle layer [52] with a bicubic layer, which we refer to as Variant 1. To validate the effectiveness of LFAUM, we introduce two additional variants. In Variant 2, we remove the operation of learning receptive field offsets in LFAUM and perform upsampling solely based on local features, scale information, and position information to generate dynamic convolutional

Fig. 7. Visualization result of the proposed method on real-world images.

TABLE V
PSNR (dB) RESULTS ACHIEVED BY OUR NETWORK WITH DIFFERENT SETTINGS ON URBAN100 BENCHMARK.

Methods	SLFAM			LFAUM		×1.7	×2	×2.95	×3.1	$\frac{\times 1.3}{\times 3.9}$	$\frac{\times 2}{\times 3.3}$
	Dy-conv	Local feature	Dep-conv	Local feature	Def-conv						
RCAN [38]	✗	✗	✗	✗	✗	35.17	33.18	29.06	28.84	28.79	29.85
Variant 1	✗	✗	✗	Bicubic		34.96	32.90	28.35	27.97	27.88	28.71
Variant 2	✗	✗	✗	✓	✗	35.21	33.33	29.20	28.82	28.80	29.94
Variant 3	✗	✗	✗	✓	✓	35.30	33.40	29.31	28.92	28.89	30.01
Variant 4	✓	✗	✗	✓	✓	35.37	33.46	29.38	29.00	28.94	30.06
Variant 5	✓	✓	✗	✓	✓	35.41	33.45	29.42	29.06	28.99	30.10
Variant 6	✓	✓	✓	✓	✓	35.43	33.51	29.44	29.10	29.05	30.12

kernels. In Variant 3, we add deformable convolution operation and utilize the complete LFAUM for upsampling reconstruction.

Table V shows that when using bicubic upsampling, the PSNR values are relatively low. By gradually adding scale and position information, local feature information, and feature offset information, the performance is significantly improved (*e.g.,* 28.92 dB vs. 27.97 dB for ×3.1 SR). This is because our upsampling layer can perceive scale-aware feature-adaptive dynamic filters, while bicubic method uses fixed filters that cannot adapt well to images with different scales and contents. Especially with the addition of deformable convolution, the reconstruction ability of the model in the complex local region is greatly enhanced, and the PSNR is also significantly improved.

b) For SLFAM: we introduce three additional variants while keeping LFAUM as the upsampling layer. In Variant 4, we use dynamic convolution to generate scale-aware convolutional kernels and apply them to the features. It is important to note that the input for generating dynamic convolutional kernels only includes scale information and does not include local features of the image. Additionally, there is only one generated dynamic convolutional kernel instead of separately generating depthwise kernels and pointwise kernels, which means the operation of depthwise separable convolution is not included. For Variant 5, we introduce local feature information into the attention network that generates dynamic convolutional kernels, to investigate the importance of the network's adaptation to local features. Finally, we use the complete SLFAM as Variant 6, which incorporates the operation of depthwise separable convolution based Variant 5. From Table V, it can be observed that both the scale-aware dynamic convolution and the utilization of local features contribute to performance improvement (*e.g.,* 30.01 dB vs. 30.10 dB for ×2/×3.3 SR).

The reason why Variant 5 outperforms Variant 4 is that Variant 4 does not consider the differences in local features between different regions of the image. The region with simple structure and the region with complex details obviously have different convolutional kernels, which should be distinguished in feature extraction to adapt to the local information of the image. By utilizing our scale-aware feature-adaptive convolutional kernels, the network can focus on image content based on scale information and local features, resulting in the generation of fine details. Variant 6, with the inclusion of depthwise separable convolution, increases the complexity

of the network as both depthwise and pointwise kernels are generated dynamically, allowing attention to be focused on both channel and pixel levels. Although the number of parameters increases compared to generating a single convolutional kernel due to the two attention branches, the overall number of parameter decreases significantly when performing subsequent convolutions with depthwise separable convolution, resulting in a reduction in overall parameter count.

TABLE VI
PSNR (dB) RESULTS ACHIEVED BY PROPOSED METHOD WITH DIFFERENT SETTINGS OF DYNAMIC CONVOLUTION WEIGHTS ON SET5. THE RUNNING TIME IS AVERAGED OVER B100 FOR ×4 SR.

Branches	Params.	Time	×1.6 ×2.4 ×3.1	$\frac{\times 1.5}{\times 4}$ $\frac{\times 1.5}{\times 3.5}$ $\frac{\times 1.6}{\times 3.05}$
1	17.00M	0.10s	40.28 36.40 34.24	33.96 35.07 35.75
2	17.02M	0.11s	40.72 36.73 34.67	34.55 35.57 36.18
4	17.08M	0.11s	40.73 36.73 34.69	34.62 35.61 36.20
8	17.20M	0.12s	40.73 36.75 34.68	34.64 35.64 36.21

2) Discussion on the Dynamic Convolution Weights: Scale-aware dynamic convolution provides adaptive and refined convolution kernel selection for feature extraction. In this section, we first study the effect of the number of dynamic convolution weights on the model performance. Then we explore the relationship between the dynamic convolution weights and the reconstruction scale factors.

a) Number of Dynamic Convolution Weights: Dynamic convolution generates multiple dynamic weights through attention branch. To explore the impact of dynamic convolution weights, we compare the effect of different numbers of weights on the network in Table VI. When only one single weight exists, the dynamic convolution is downgraded to a regular convolution with static kernels, which cannot deal with multiple scale factors and image contents effectively. Therefore, the performance of Variant 1 decreases from 40.73 dB to 40.28 dB on ×1.6 SR. As the number of dynamic convolution weights increases, the network shows improved performance on symmetric scale factors and more significant effects on asymmetric scale factors. Particularly, there is significant improvement for highly asymmetric scale factors (*e.g.,* 34.65 dB vs. 33.96 dB on ×1.5/×4 SR). To achieve a preferable trade-off between model size and performance, we choose four dynamic convolution weights as the default setting since having more than four attention branches does not bring significant performance improvement.

b) Relationship with Scale Factors: In Fig. 8, we show the relationship between dynamic attention weights (*i.e.*, depth-wise convolution weight π_{dw} and point-wise convolution weight π_{dw}) and scale factors and image content respectively. We set the attention branch to four and choose the average result on Set5 [49] as a reference. From this, we get the four weights π^1_{dw}, π^2_{dw}, π^3_{dw} and π^4_{dw} of the depth-wise branch and π^1_{pw}, π^2_{pw}, π^3_{pw} and π^4_{pw} of the point-wise branch. When the upsampling scale is small, it is mainly π^1_{dw} and π^1_{pw} that are activated, which values are close to 1. The remaining weights have values close to 0, which is equivalent to a conventional convolution. This means that conventional static convolution can meet the requirements of the model to achieve small-scale SR tasks. As the scale factor increases, the values of weight π^1_{dw} and π^1_{pw} decrease, while the values of weight π^2_{dw}, π^3_{dw}, π^4_{dw}, π^2_{pw}, π^3_{pw} and π^4_{pw} gradually increase, and more convolution kernels are activated. This indicates that the larger the scale factor, the stronger the dependence of the model on the scale-aware dynamic convolution. Because the scale-aware dynamic convolution fully considers the impact of scale on the model, and enhances the adaptability of the network to large-scale upsampling. These results validate the effectiveness of the dynamic weight generator.

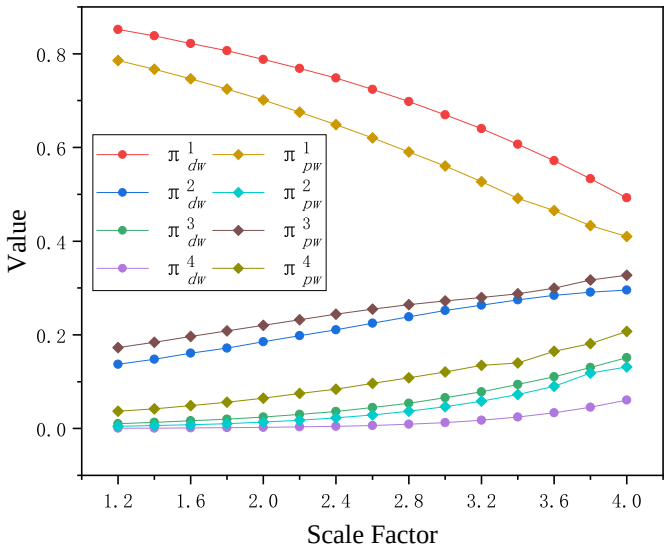


Fig. 8. Relationship between dynamic weights and scale factors on Set5 dataset.

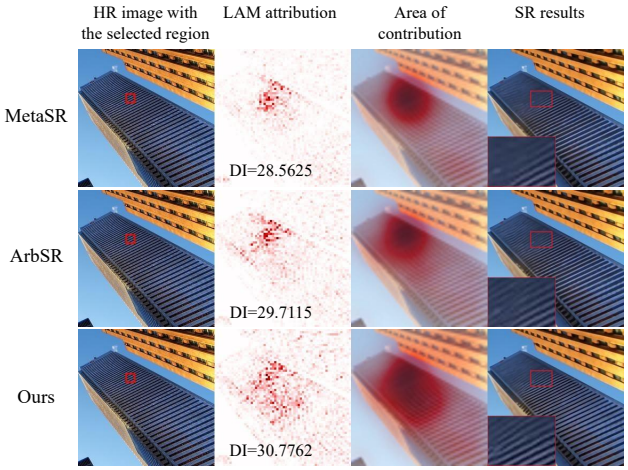


Fig. 9. LAM [58] results and visual comparison for different methods.

G. Attribution Analysis

In order to verify whether our model activates more pixels, we conduct attribution analysis of different methods through local attribution maps (LAM) [58] and diffusion index (DI), as shown in Fig. 9. LAM attribution represents the contribution of each pixel to the SR result of a given patch. DI is a statistical dispersion measure, and the larger the DI value, the more pixels contribute to the SR result. According to the results, our method activates more pixels than MetaSR [16]

and ArbSR [8]. The reason is that our module integrates a variety of information such as scale factor, local feature, pixel projection position and receptive field offset into the network, enabling the network to make full use of image information for reconstruction, significantly increasing the complexity of the network. This conclusion confirms the effectiveness of our network.

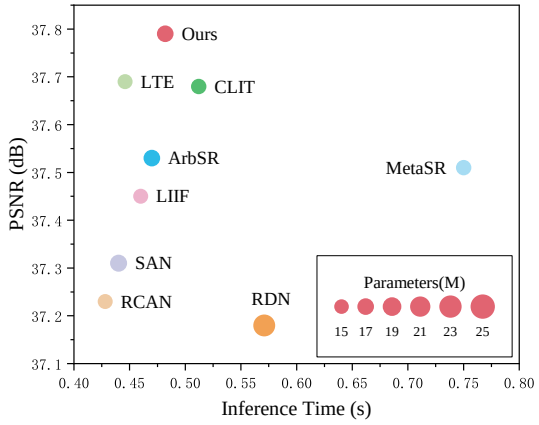


Fig. 10. Inference time and number of parameters for SR algorithms.

H. Investigation of Model Complexity and Inference Time

In this section, we evaluate the computational complexity of the models based on their inference time and number of parameters. Fig. 10 compares the running time (s) and PSNR (dB) of different algorithms. The running time is the average time taken to reconstruct images on $\times 1.5$ scale using the Set14 dataset. Fig. 10 compares the numbers of parameters (M) and PSNR of different algorithms. From these two figures, it can be observed that our model has an average running time that is 0.3s lower than CLIT [56], and slightly higher than ArbSR [8] and LTE [46], which overall ensures the speed of the network. In terms of parameter count, our network introduces two sets of pluggable modules based on RCAN [38], resulting in a parameter increase of 0.15M compared to ArbSR [8]. Although there is a slight increase in parameter count, it does not significantly affect the running time, while achieving a noticeable improvement in PSNR. Overall, our method demonstrates certain advantages in terms of computational complexity.

V. Conclusion

To activate more information for better reconstruction, we propose two plug-in high-performance modules for arbitrary-scale super-resolution tasks. Both modules can be easily combined with existing fixed-scale super-resolution networks to produce results with non-integer and asymmetric scale factors. SLFAM based on dynamic convolution can generate dynamic kernel according to scale factors and local structure, and extract adaptive features. LFAUM based on depth separable convolution generates adaptively using convolution kernel by fusing scale and local feature information, and learns offset to make the upsampling receptive field change direction and range adaptively according to image texture, thus generating SR image with high adaptability to image content. Extensive experimental results have demonstrated to verify the effectiveness of proposed module, and the performance advantage of our module is significant in the case of similar complexity.

References

[1] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[2] H. Chen, X. He, L. Qing, Y. Wu, C. Ren, R. E. Sheriff, and C. Zhu, "Real-world single image super-resolution: A brief review," *Information Fusion*, vol. 79, pp. 124–145, 2022.

[3] S. Anwar, S. Khan, and N. Barnes, "A deep journey into super-resolution: A survey," *ACM Computing Surveys*, vol. 53, no. 3, pp. 1–34, 2020.

[4] B. B. Moser, F. Raue, S. Frolov, S. Palacio, J. Hees, and A. Dengel, "Hitchhiker's guide to super-resolution: Introduction and recent advances," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 8, pp. 9862–9882, 2023.

[5] F. Li, Y. Wu, H. Bai, W. Lin, R. Cong, and Y. Zhao, "Learning detail-structure alternative optimization for blind super-resolution," *IEEE Transactions on Multimedia*, vol. 25, pp. 2825–2838, 2023.

[6] H. Wang, X. Chen, B. Ni, Y. Liu, and J. Liu, "Omni aggregation networks for lightweight image super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 22 378–22 387.

[7] Y. Liu, S. Wang, J. Zhang, S. Wang, S. Ma, and W. Gao, "Iterative network for image super-resolution," *IEEE Transactions on Multimedia*, vol. 24, pp. 2259–2272, 2022.

[8] L. Wang, Y. Wang, Z. Lin, J. Yang, W. An, and Y. Guo, "Learning a single network for scale-arbitrary super-resolution," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 4801–4810.

[9] M.-I. Georgescu, R. T. Ionescu, A.-I. Miron, O. Savencu, N.-C. Ristea, N. Verga, and F. S. Khan, "Multimodal multi-head convolutional attention with various kernel sizes for medical image super-resolution," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 2195–2205.

[10] H. Yang, Z. Wang, X. Liu, C. Li, J. Xin, and Z. Wang, "Deep learning in medical image super resolution: a review," *Applied Intelligence*, pp. 1–26, 2023.

[11] K. Chen, W. Li, S. Lei, J. Chen, X. Jiang, Z. Zou, and Z. Shi, "Continuous remote sensing image super-resolution based on context interaction in implicit function space," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–16, 2023.

[12] H. Wu, N. Ni, and L. Zhang, "Learning dynamic scale awareness and global implicit functions for continuous-scale super-resolution of remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–15, 2023.

[13] G. Gao, L. Tang, F. Wu, H. Lu, and J. Yang, "JDSR-GAN: Constructing an efficient joint learning network for masked face super-resolution," *IEEE Transactions on Multimedia*, vol. 25, pp. 1505–1512, 2023.

[14] A. Agarwal, N. Ratha, M. Vatsa, and R. Singh, "Impact of super-resolution and human identification in drone surveillance," in *2021 IEEE/CVF International Workshop on Information Forensics and Security (WIFS)*, 2021, pp. 1–6.

[15] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 136–144.

[16] X. Hu, H. Mu, X. Zhang, Z. Wang, T. Tan, and J. Sun, "Meta-SR: A magnification-arbitrary network for super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1575–1584.

[17] C. Lemke, M. Budka, and B. Gabrys, "Metalearning: a survey of trends and technologies," *Artificial Intelligence Review*, vol. 44, pp. 117–130, 2015.

[18] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu, "Dynamic convolution: Attention over convolution kernels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 030–11 039.

[19] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2017, pp. 764–773.

[20] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable convnets v2: More deformable, better results," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9308–9316.

[21] Y. Romano, M. Protter, and M. Elad, "Single image interpolation via adaptive nonlocal sparsity-based modeling," *IEEE Transactions on Image Processing*, vol. 23, no. 7, pp. 3085–3098, 2014.

[22] S. Zhu, Z. He, S. Liu, and B. Zeng, "Mmse-directed linear image interpolation based on nonlocal geometric similarity," *IEEE Signal Processing Letters*, vol. 24, no. 8, pp. 1178–1182, 2017.

[23] W. Dong, L. Zhang, R. Lukac, and G. Shi, "Sparse representation based image interpolation with nonlocal autoregressive modeling," *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1382–1394, 2013.

[24] J. Jiang, X. Ma, C. Chen, T. Lu, Z. Wang, and J. Ma, "Single image super-resolution via locally regularized anchored neighborhood regression and nonlocal means," *IEEE Transactions on Multimedia*, vol. 19, no. 1, pp. 15–26, 2016.

[25] H. Chen, X. He, L. Qing, and Q. Teng, "Single image super-resolution via adaptive transform-based nonlocal self-similarity modeling and learning-based gradient regularization," *IEEE Transactions on Multimedia*, vol. 19, no. 8, pp. 1702–1717, 2017.

[26] G. Chantas, S. N. Nikolopoulos, and I. Kompatsiaris, "Heavy-tailed self-similarity modeling for single image super resolution," *IEEE Transactions on Image Processing*, vol. 30, pp. 838–852, 2020.

[27] J. Sun, Z. Xu, and H.-Y. Shum, "Image super-resolution using gradient profile prior," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[28] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang, "Coupled dictionary training for image super-resolution," *IEEE Transactions on Image Processing*, vol. 21, no. 8, pp. 3467–3478, 2012.

[29] K. Zhang, X. Gao, D. Tao, and X. Li, "Single image super-resolution with non-local means and steering kernel regression," *IEEE Transactions on Image Processing*, vol. 21, no. 11, pp. 4544–4556, 2012.

[30] W. Li, J. Li, G. Gao, W. Deng, J. Zhou, J. Yang, and G.-J. Qi, "Cross-receptive focused inference network for lightweight image super-resolution," *IEEE Transactions on Multimedia*, 2023.

[31] F. Li, Y. Wu, H. Bai, W. Lin, R. Cong, and Y. Zhao, "Learning detail-structure alternative optimization for blind super-resolution," *IEEE Transactions on Multimedia*, vol. 25, pp. 2825–2838, 2023.

[32] Y. Zhang, L. Dong, H. Yang, L. Qing, X. He, and H. Chen, "Weakly-supervised contrastive learning-based implicit degradation modeling for blind image super-resolution," *Knowledge-Based Systems*, vol. 249, p. 108984, 2022.

[33] C. Tian, Y. Xu, W. Zuo, B. Zhang, L. Fei, and C.-W. Lin, "Coarse-to-fine cnn for image super-resolution," *IEEE Transactions on Multimedia*, vol. 23, pp. 1489–1502, 2020.

[34] D. Zhang, J. Shao, Z. Liang, L. Gao, and H. T. Shen, "Large factor image super-resolution with cascaded convolutional neural networks," *IEEE Transactions on Multimedia*, vol. 23, pp. 2172–2184, 2020.

[35] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1646–1654.

[36] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 136–144.

[37] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proceedings of the IEEE Ronference on Computer Vision and Pattern Recognition*, 2018, pp. 2472–2481.

[38] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 286–301.

[39] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang, "Second-order attention network for single image super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 065–11 074.

[40] X. Chen, X. Wang, J. Zhou, Y. Qiao, and C. Dong, "Activating more pixels in image super-resolution transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 22 367–22 377.

[41] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[42] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1874–1883.

[43] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proceedings of the European Conference on Computer Vision*, 2014, pp. 818–833.

[44] Y. Fu, J. Chen, T. Zhang, and Y. Lin, "Residual scale attention network for arbitrary scale image super-resolution," *Neurocomputing*, vol. 427, pp. 201–211, 2021.

[45] Y. Chen, S. Liu, and X. Wang, "Learning continuous image representation with local implicit image function," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8628–8638.

[46] J. Lee and K. H. Jin, "Local texture estimator for implicit representation function," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1929–1938.

[47] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1251–1258.

[48] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 126–135.

[49] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proceedings of the British Machine Vision Conference*, 2012, pp. 135.1–135.10.

[50] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Curves and Surfaces: 7th International Conference*, 2012, pp. 711–730.

[51] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings of IEEE/CVF International Conference on Computer Vision*, 2001, pp. 416–423.

[52] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5197–5206.

[53] Y. Matsui, K. Ito, Y. Aramaki, A. Fujimoto, T. Ogawa, T. Yamasaki, and K. Aizawa, "Sketch-based manga retrieval using manga109 dataset," *Multimedia Tools and Applications*, vol. 76, pp. 21 811–21 838, 2017.

[54] Z. 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*, vol. 13, no. 4, pp. 600–612, 2004.

[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, 2015, pp. 1–15.

[56] H.-W. Chen, Y.-S. Xu, M.-F. Hong, Y.-M. Tsai, H.-K. Kuo, and C.-Y. Lee, "Cascaded local implicit transformer for arbitrary-scale super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 18 257–18 267.

[57] J. Cai, H. Zeng, H. Yong, Z. Cao, and L. Zhang, "Toward real-world single image super-resolution: A new benchmark and a new model," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3086–3095.

[58] J. Gu and C. Dong, "Interpreting super-resolution networks with local attribution maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9199–9208.