Latent Modulated Function for Computational Optimal Continuous Image Representation

Supplementary Material

We organize our supplementary material as follows. In Section 1, we provide additional implementation details and comprehensive network architectures for our proposed LMF-based ASSR methods. In Section 2, we conduct an in-depth efficiency analysis of LMF-based ASSR methods and the original ASSR methods. Section 3 showcases the effectiveness of CMSR by providing additional details and results. Finally, Section 4 delves into more visual results for various ASSR scenarios.

1. More Implementation Details

In Table 1, we present the detailed network structures of our LMF-based ASSR methods, namely LM-LIIF, LM-LTE, and LM-CiaoSR. For reference, we also provide the network architectures of the original INR-based ASSR methods, namely LIIF [2], LTE [3], and CiaoSR [1]. In the table, the notation $MLP = [580 \times 256, 3 \times (256 \times 256), 256 \times 3]$ represents an MLP architecture. Firstly, it includes an input linear layer with input and output dimensions of 580 and 256, respectively. Additionally, there are three hidden linear layers, each with both input and output dimensions of 256. Finally, there is an output linear layer with input and output dimensions of 256 and 3, respectively. Regarding the activation function, we utilize the ReLU activation after every linear layer, except for the output linear layer in the MLP, which does not use any activation function. The notation $Conv = [64 \times 256], ks = 3$ represents a convolutional layer with input and output features of 64 and 256 dimensions, respectively, and a kernel size of 3.

Local ensemble is proposed in LIIF [2], which ensures continuous prediction by merging the four predictions from the top-left, top-right, bottom-left, and bottom-right latent codes. Both our LMF-based ASSR methods and the reference methods utilize local ensemble in HR space:

$$I(x^*_{i,j}) = \sum_{t\in\{tl,tr,bl,br\}} \frac{A(v^*_{t'}, x^*_{i,j})}{A_{R^*}} f_{\theta_r, m^*_{t'}}(z^*_{t'}, x^*_{i,j}), \quad (1)$$

where $x^*_{i,j}$ is any query coordinate closest to z^*, and v^*_t is the coordinate of z^*_t. z^*_t is one of the latent codes at the top-left (tl), top-right (tr), bottom-left (bl), bottom-right (br) corners center around z^*. $A(v^*_{t'}, x^*_{i,j})$ refers to the area of the rectangle between the coordinates $v^*_{t'}$ and $x^*_{i,j}$, where t' is diagonally opposite to t (*i.e*., (1,1) to (-1,-1), (-1,1) to (1,-1)). A_{R^*} represents the area of the rectangle R^* whose vertices are $\{tl, tr, bl, br\}$.

LIIF also introduces feature unfolding [2], which concatenates the nearby 3×3 feature vectors as the latent code:

$$z^* = Concat(\{z^*_{k,l}\}_{k,l\in\{-1,0,1\}}), \quad (2)$$

where $z^*_{k,l}$ is one of the 3×3 feature vectors center around the latent code z^*. Both LIIF and CiaoSR require the use of 3×3 feature unfolding in HR space, while LM-LIIF and LM-CiaoSR only require 3×3 and 2×2 feature unfolding in latent space, respectively. In contrast, both LTE and LM-LTE replace the feature unfolding strategy with the Local Texture Estimator [3], This estimator consists of a frequency convolutional layer, a coefficient convolutional layer, and an MLP for phase.

2. Efficiency Analysis

In this section, we present the computational analysis of LIIF and LM-LIIF. These two methods are representative as they focus on the fundamentals of continuous image representation. We analyze upsampling an $h \cdot w \cdot 3$ input image with a scale factor s. Before decoding, the encoder generates $h \cdot w \cdot 64$ feature maps using the input image. To quantify the computational complexity of LIIF, we can formulate it as the number of multiplications used by LIIF:

$$\begin{aligned} Mul_{\text{LIIF}} &= 4(d_{in}d_h + (k-2)d_h^2 + d_h d_{out})s^2hw \\ &= 1383424s^2hw, \end{aligned} \quad (3)$$

where $d_{in} = 580$ denotes the dimension of the MLP input, which consists of coordinates, cells, and the latent code using 3×3 feature unfolding. $d_h = 256$ represents the hidden dimension of the decoding MLP, while $d_{out} = 3$ represents the dimension of the output pixel value. The depth of the decoding MLP is denoted by $k = 5$.

To accomplish the same upsampling task, the number of multiplications used by LM-LIIF can be formulated as:

$$\begin{aligned} Mul_{\text{LM-LIIF}} =& (d_{in}d_l + (k_l - 1)d_l^2)hw + \\ & 4(d_c d_r + (k_r - 2)d_r^2 + d_r d_{out})s^2hw \\ =& 163488hw + 6592s^2hw, \end{aligned} \quad (4)$$

where $d_{in} = 578$ represents the dimension of the latent MLP input, which consists of cells and the latent code using 3×3 feature unfolding. $d_c = 20$ represents the dimension of the render MLP input, which consists of the compressed latent code, coordinates, and cells. The hidden dimension of the latent MLP and the render MLP are denoted by $d_l = 208$ and $d_r = 16$, respectively. The depth of the latent MLP and the render MLP are represented by $k_l = 2$ and $k_r = 7$, respectively.

Table 1. The network architectures of LMF-based ASSR methods and the reference methods. * indicates the use of latent modulations.

Decoder	Params	Latent modulation	Latent space	Render space
LIIF	355K	$\times$	$\times$	$MLP = [580 \times 256, 3 \times (256 \times 256), 256 \times 3]$
LM-LIIF	170K	192	$MLP_l = [578 \times 208, 208 \times 208]$	$MLP_r = [20 \times 16^*, 5 \times (16 \times 16^*), 16 \times 3]$
LTE	506K	$\times$	$Conv_{freq} = [64 \times 256], ks3$, $Conv_{coef} = [64 \times 256], ks3$, $MLP_{phase} = [2 \times 128]$	$MLP_r = [256 \times 256, 2 \times (256 \times 256), 256 \times 3]$
LM-LTE	278K	256	$Conv_{freq} = [64 \times 128], ks3$, $Conv_{coef} = [64 \times 128], ks3$, $MLP_{phase1} = [2 \times 64]$, $MLP_l = [128 \times 288, 288 \times 288]$	$MLP_{phase2} = [2 \times 8]$, $MLP_r = [16 \times 16^*, 7 \times (16 \times 16^*), 16 \times 3]$
CiaoSR-1.4M	1.429M	$\times$	Non-local attention module	$MLP_k = [580 \times 256, 3 \times (256 \times 256), 256 \times 576]$, $MLP_v = [644 \times 256, 3 \times (256 \times 256), 256 \times 640]$, $MLP_q = [640 \times 256, 3 \times (256 \times 256), 256 \times 3]$
CiaoSR-3.1M	3.065M	$\times$	Non-local attention module	$MLP_k = [1624 \times 256, 3 \times (256 \times 256), 256 \times 1620]$, $MLP_v = [1804 \times 256, 3 \times (256 \times 256), 1800 \times 3]$, $MLP_q = [1800 \times 256, 3 \times (256 \times 256), 256 \times 3]$
LM-CiaoSR	753K	256	Non-local attention module, $MLP_k = [260 \times 256, 256 \times 256]$, $MLP_v = [324 \times 256, 256 \times 320]$, $MLP_q = [576 \times 256, 256 \times 256]$, $MLP_l = [320 \times 288, 288 \times 288]$	$MLP_r = [34 \times 16^*, 7 \times (16 \times 16^*), 16 \times 3]$

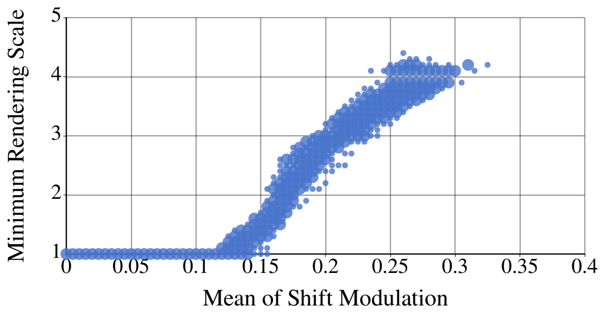

Figure 1. The positive correlation between the means of shift modulation and the minimum rendering scale factors in LM-LTE.

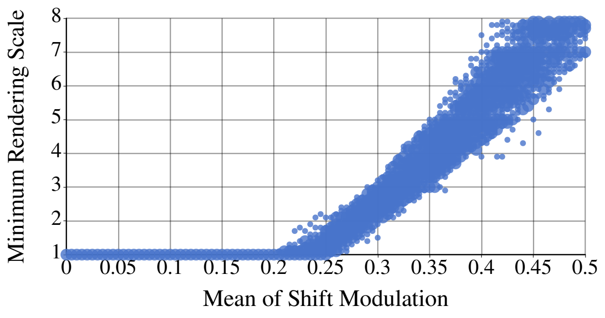

Figure 2. The positive correlation between the means of shift modulation and the minimum rendering scale factors in LM-CiaoSR.

Compare Eq. 3 and Eq. 4, it becomes apparent that our LM-LIIF requires significantly fewer multiplications than LIIF, and the reduction in multiplications $Mul_{\text{LIIF}} - Mul_{\text{LM-LIIF}}$ increases as the scale factor increases. For large scales, most of the multiplications in LM-LIIF are attributed to the latent MLP, which remains unaffected by the scale factor. Even for identity mapping ($\times 1$ SR), our LM-LIIF reduces the number of multiplications by over 87.7%.

3. Details and Results on CMSR

In Figures 1 and 2, we visualize the positive correlation between the means of shift modulation and the minimum rendering scale factors in two different methods, LM-LTE and LM-CiaoSR. Specifically, we uniformly sample 100 means of shift modulation from the range of $[0, 0.5]$ and measure their corresponding minimum scale factors using the LR images from the training set. The results clearly indicate a positive correlation between the means of shift modulation and the minimum scale factors in both LM-LTE and LM-CiaoSR methods. An intriguing observation is that the minimum scale factors in LM-CiaoSR are relatively larger than those in LM-LTE, suggesting superior SR performance and generalization of LM-CiaoSR over LM-LTE. These results provide valuable and comprehensive insights into the effectiveness of latent modulation and the underlying mechanisms and performance characteristics of INR-based ASSR methods.

In Algorithm 1, we present the detailed algorithm for creating the Scale2Mod table, denoted as $T = \{s_0 : [m_{min}^{s_0}, m_{max}^{s_0}], \dots, s_K : [m_{min}^{s_K}, m_{max}^{s_K}]\}$. The purpose of this table T is to map a given scale factor s_k to the range of means of shift modulation $[m_{min}^{s_k}, m_{max}^{s_k}]$. For any input pixel whose mean of shift modulation m satisfies $m_{min}^{s_k} \leq m \leq m_{max}^{s_k}$, we render this pixel with its minimum scale

Algorithm 1: Creating Scale2Mods table for LMF

```
Input : LR images $\{I_0, \dots, I_J\}$, MSE threshold $a$,
        common scale factors $\{s_0, \dots, s_K\}$,
        means of shift modulation
        $\{0, u, 2u, \dots, 1\}$ sampled with interval $u$
Output: Scale2Mods table $T = \{s_0 : [m^{s_0}_{min}, m^{s_0}_{max}], \dots, s_K : [m^{s_K}_{min}, m^{s_K}_{max}]\}$
1  for $I_j \in \{I_0, \dots, I_J\}$ do
2      $I^{s_K}_j = Render_{s_K}(I_j)$;
3      Compute means of shift modulation $M_j$ for $I_j$;
4      for $s_k \in \{s_0, \dots, s_K\}$ do
5          $I^{s'_K}_j = Bilinear_{\frac{s_K}{s_k}}(Render_{s_k}(I_j))$;
6          $[m^{s_k}_{min}, m^{s_k}_{max}] = T(s_k)$;
7          for $m_l \in \{0, u, 2u, \dots, 1\}$ do
8              Create a filter
                 $Filter(I) : m_l - \frac{u}{2} \leq M \leq m_l + \frac{u}{2}$;
9              $Error_{k,l} =$
                 $MSE(Filter(I^{s_K}_j), Filter(I^{s'_K}_j))$;
10             if $Error_{k,l} \leq a$ then
11                 $m^{s_k}_{max} = Min(m^{s_k}_{max}, m_l)$;
12                 $m^{s_{k+1}}_{min} = m^{s_k}_{max}$;
13             else
14                 break;
15             end
16             $T(s_k) = [m^{s_k}_{min}, m^{s_k}_{max}]$;
17         end
18     end
19 end
20 return $T$;
```

factor s_k by querying the table. To create a Scale2Mod table with a specified MSE threshold a, we follow the steps outlined below. For each LR image $I_j \in \{I_0, \dots, I_J\}$ from the training set, we render I_j with the maximum scale s_K to obtain the fully rendered image $I^{s_K}_j$. Also, we compute the means of shift modulation M_j corresponding to I_j. Next, for each scale factor $s_k \in \{s_0, \dots, s_K\}$ and each mean of shift modulation $m_l \in \{0, u, 2u, \dots, 1\}$, we calculate the MSE between the pixels in the fully rendered image $I^{s_K}_j$ and the pixels in the image rendered with s_k followed by bilinear interpolation with $\frac{s_K}{s_k}$. Notably, we only compute the MSE for any pixel whose corresponding mean of shift modulation satisfies the condition $Abs(m - m_l) \leq \frac{u}{2}$. If the MSE result is less than or equal to the MSE threshold a, we update the $m^{s_k}_{max}$ and $m^{s_{k+1}}_{min}$ in T using the current mean of shift modulation m_l. This process is repeated for each LR image in the training set, enabling us to create a generalized Scale2Mod table T.

In Algorithm 2, we demonstrate the detailed algorithm for CMSR. Given an LR image I_{LR}, a target upsampling

Algorithm 2: Controllable Multi-Scale Rendering

```
Input : LR image $I_{LR}$, target scale factor $s_{target}$,
        Scale2Mods table $T = \{s_0 : [m^{s_0}_{min}, m^{s_0}_{max}], \dots, s_K : [m^{s_K}_{min}, m^{s_K}_{max}]\}$
Output: SR image $I_{SR}$
1  Compute means of shift modulation $M_{LR}$ for $I_{LR}$;
2  Initialize $s_{-1} = 1, I_{s_{-1}} = I_{LR}$;
3  for $s_k \in \{s_0, \dots, s_K\}$ do
4      if $s_k > s_{target}$ then
5          $I_{SR} = Bilinear_{\frac{s_{target}}{s_{k-1}}}(I_{s_{k-1}})$;
6          Create a filter $Filter(I) : M \geq m^{s_{k-1}}_{max}$;
7          $Filter(I_{SR}) =$
             $Render_{s_{target}}(Filter(I_{LR}))$;
8          break;
9      end
10     $I_{s_k} = Bilinear_{\frac{s_k}{s_{k-1}}}(I_{s_{k-1}})$;
11     $[m^{s_k}_{min}, m^{s_k}_{max}] = T(s_k)$;
12     Create a filter $Filter(I) : m^{s_k}_{min} \leq M \leq m^{s_k}_{max}$;
13     $Filter(I_{s_k}) = Render_{s_k}(Filter(I_{LR}))$;
14     if $\forall m \in M_{LR}, m <= m^{s_k}_{max}$ then
15         $I_{SR} = Bilinear_{\frac{s_{target}}{s_k}}(I_{s_k})$;
16         break;
17     end
18 end
19 return $I_{SR}$;
```

scale factor s_{target}, and a Scale2Mods table T, we proceed as follows. Firstly, we compute the means of shift modulation M_j corresponding to I_j, and initialize s_{-1} as 1, $I_{s_{-1}}$ as I_{LR}. For each scale factor $s_k \in \{s_0, \dots, s_K\}$, we check if s_k is greater than s_{target}. If it is, we upsample the result from the last scale s_{k-1} using bilinear interpolation with $\frac{s_{target}}{s_{k-1}}$. We then replace the unrendered pixels in this bilinear upsampled result with the corresponding pixels rendered with s_{target}. On the other hand, if s_k is less than or equal to s_{target}, we upsample the last result $I_{s_{k-1}}$ using bilinear interpolation with $\frac{s_k}{s_{k-1}}$ to obtain the current result I_{s_k}. Next, we query the Scale2Mods table T to obtain the range of means of shift modulation $[m^{s_k}_{min}, m^{s_k}_{max}]$. For the current scale s_i, we only need to render the pixels whose means of shift modulation fall within this range $[m^{i}_{min}, m^{i}_{max}]$. We repeat the above process iteratively until we obtain the final SR result or until all pixels have been rendered by LMF. This CMSR algorithm ensures that LMF takes into account the content-awareness of the LR image, allowing for effective and efficient SR with satisfactory visual quality.

For an in-depth analysis of the computational savings afforded by CMSR, Table 2 provides the MACs results on DIV2K validation set of the latent MLP and the render MLP with or without the final CMSR. EDSR-b [5] is utilized as

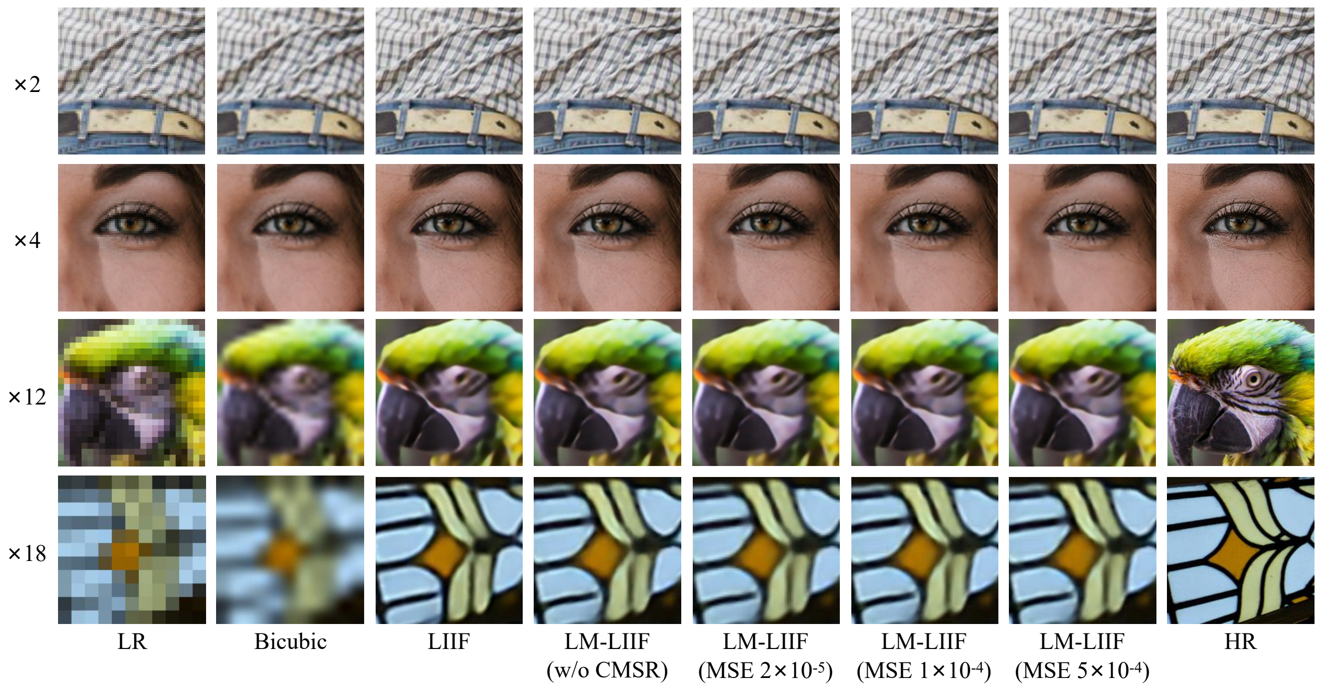

Figure 3. Qualitative comparison for using Scale2Mod tables with different MSE thresholds in CMSR. All of the ASSR methods use EDSR-b as the encoder.

Table 2. Ablation study (MACs results on DIV2K validation set) on the latent MLP and the render MLP with and w/o CMSR.

Method	Network	CMSR	×2	×4	×8	×16
LIIF	Decoder	-	3.567T	3.567T	3.567T	3.567T
LM-LIIF	Latent MLP	-	107.92G	26.98G	6.73G	1.67G
LM-LIIF	Render MLP	×	17.41G	17.41G	17.37G	17.23G
LM-LIIF	Render MLP	✓	16.30G	13.64G	12.41G	3.48G

the encoder. The results reveal that, particularly at scales 8 and 16, the computational cost is primarily attributable to the render MLP, and CMSR can significantly reduces the computations for the render MLP at these scales.

In Table 3, we further investigate the effectiveness of using the means of shift modulation as the indicators of signal complexity. Here, we manually adjust the means of shift modulation by ± 0.2 when used in CMSR. Utilizing EDSR-b [5] as the encoder and LM-LIIF as the decoder, the results validate that querying rendering scales based on precise means of shift modulation achieves an optimal balance between PSNR performance and computational efficiency.

In Figure 3, we present the qualitative comparison of using Scale2Mod tables at varying MSE thresholds in CMSR. Specifically, we experiment with thresholds at 2×10^{-5}, 1×10^{-4}, and 5×10^{-4}. All of the ASSR methods utilize EDSR-b [5] as the encoder, and our LMF-based ASSR methods utilize LM-LIIF as the decoder. As depicted in the figure, setting the MSE threshold of 2×10^{-5} in CMSR

Table 3. Ablation study (PSNR/MACs of the render MLP on DIV2K validation set) on the means of shift modulation (referred to as "Mean") in CMSR.

MSE thresh.	Query	×2	×4	×8	×16
1×10^{-4}	Mean	34.62/ **13.63G**	28.98/ **11.71G**	25.39/ **8.35G**	**22.62/ 2.50G**
2×10^{-5}	Mean - 0.2	34.53/ 14.88G	28.98/ 16.04G	**25.40/** 14.78G	**22.62/** 2.68G
2×10^{-5}	Mean + 0.2	**34.65/** 17.41G	**28.99/** 17.40G	**25.40/** 15.45G	**22.62/** 4.28G
2×10^{-5}	Mean	**34.65/** 16.30G	**28.99/** 13.64G	**25.40/** 12.41G	**22.62/** 3.48G

yields the same visual quality as LM-LIIF without CMSR. Notably, further increasing the MSE thresholds to 1×10^{-4} and 5×10^{-4} maintains satisfactory visual quality while significantly reducing the computational cost.

4. More Visual Results

In this section, we demonstrate additional visual results obtained from test sets to showcase the effectiveness of our LMF-based ASSR methods. Figure 4 illustrates the application of ASSR methods on the challenging task of extreme large-scale SR, specifically ×30 SR. For all the ASSR methods showcased, SwinIR [4] is used as the encoder. As depicted in the figure, LMF, as a continuous image representation, excels at producing sharp edges and realistic tex-

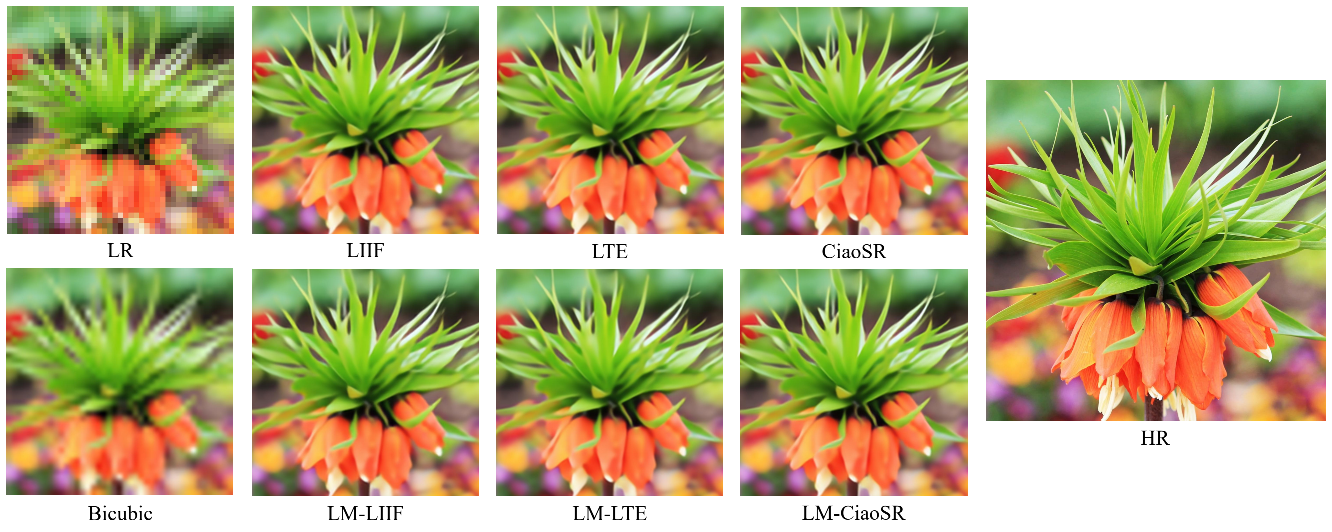

Figure 4. Qualitative comparison for $\times 30$ SR. All of the ASSR methods use SwinIR as the encoder.

tures when dealing with such extreme scales, outperforming the bicubic interpolation method by a significant margin.

In Figure 5 and Figure 6, we present additional qualitative comparisons for ASSR using EDSR-b [5] and RDN [6] as the encoders, respectively. As illustrated in the figures, our LMF-based ASSR methods achieve visual quality comparable to that of the original ASSR methods in most cases, regardless of the encoder used.

References

[1] Jiezhang Cao, Qin Wang, Yongqin Xian, Yawei Li, Bingbing Ni, Zhiming Pi, Kai Zhang, Yulun Zhang, Radu Timofte, and Luc Van Gool. Ciaosr: Continuous implicit attention-in-attention network for arbitrary-scale image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 1

[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. 1

[3] 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. 1

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

[5] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 136–144, 2017. 3, 4, 5

[6] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2472–2481, 2018. 5

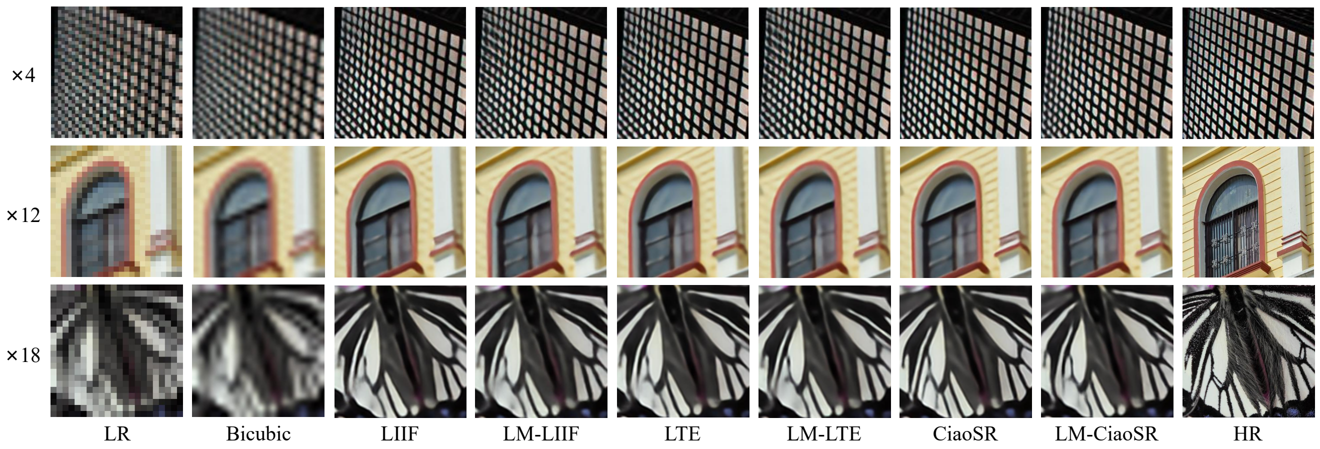

Figure 5. Qualitative comparison for ASSR. All of the ASSR methods use EDSR-b as the encoder.

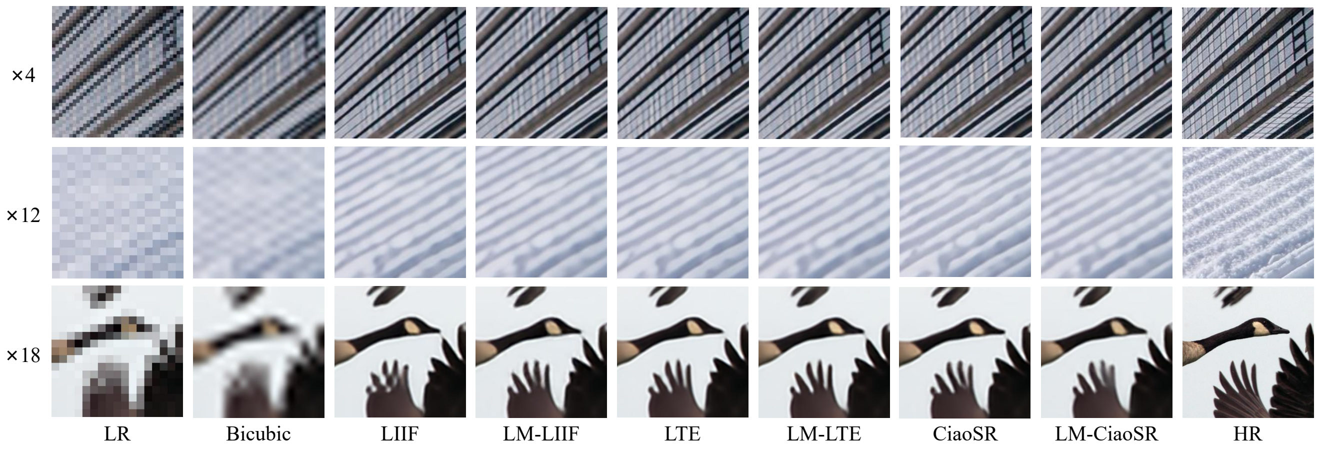

Figure 6. Qualitative comparison for ASSR. All of the ASSR methods use RDN as the encoder.