Efficient Look-Up Table from Expanded Convolutional Network for Accelerating Image Super-resolution

Kai Yin, Jie Shen*
University of Electronic Science and Technology of China
yinkai@std.uestc.edu.cn, sjie@uestc.edu.cn

Abstract
The look-up table (LUT) has recently shown its practicability and effectiveness in super-resolution (SR) tasks due to its low computational cost and hardware independence. However, most existing methods focus on improving the performance of SR, neglecting the demand for high-speed SR on low-computational edge devices. In this paper, we propose an efficient expanded convolution (EC) layer, which expands the output size of regular convolution to enlarge the receptive field (RF) indirectly. It can increase the size of the LUT corresponding to the network linearly with the increase of RF. Additionally, after introducing the EC, multiple LUTs are merged into one LUT, achieving faster running speed while maintaining SR performance. More specifically, we expand the coverage of the convolutional output so that the output at the current position covers the target position and its surroundings, forming an overlapping sliding window at the output end. We sum up the overlapping parts of the sliding window as the output, thereby achieving the effect of enlarging the RF size. Moreover, by expanding the numerical range of the accumulated results and rescaling them to \([0, 255]\), the method can mitigate the error caused by quantization output. Experiments indicate that the proposed method performs better than the baseline method and is faster than other LUT-based SR methods.

Introduction
Single image super-resolution (SISR) aims to recover high-resolution (HR) images with high-frequency image details from a single low-resolution (LR) image, which has extensive applications in medical imaging, video surveillance, satellite and aerial imaging. In the early days, interpolation-based methods were common, such as nearest neighbor interpolation, bilinear interpolation, and bicubic (Keys 1981) interpolation. These methods output the missing pixels based on the weighted average of nearby pixels at the target position. Interpolation methods have the advantages of simplicity and fast processing speed. However, interpolation methods only consider the positional information without the arrangement patterns of different pixel values, so they cannot effectively restore high-frequency signals.

Deep neural networks (DNNs) have strong fitting and nonlinear mapping capabilities. With the development of deep learning, research on using deep neural networks to solve SR problems is gradually increasing (Dong et al. 2014; Dong, Loy, and Tang 2016; Shi et al. 2016). Through its powerful nonlinear mapping ability, depth models can learn different mapping relationships for different pixel arrangements in LR images and even predict the lost high-frequency information. Therefore, deep learning methods are significantly superior to interpolation methods in restoring image details. Despite the great success of deep models, as the depth and width of the model increase, although the super-resolution effect is improved, the computational load and storage space occupied by the model are also sharply increasing. Moreover, some attention-based methods also require a large amount of memory consumption. Without high-performance hardware support, DNN-based methods are challenging to apply in practice.

A look-up table (LUT) is commonly used in low-level vision tasks by storing the outputs of complex computations in...
the LUT and retrieving these values directly without recomputing them when needed. Recently, some studies have applied LUT-based methods to SR tasks (Jo and Kim 2021; Ma et al. 2022; Li et al. 2022). These methods store the mapping relationship between input and output pixel values in LUTs and retrieve them during inference, trading the time cost of accessing memory for the time cost of complex computations, thereby accelerating the SR inference process. Since the inference time cost for these methods mainly comes from memory access, they could achieve real-time SR without specific hardware acceleration, significantly improving the practicality of SR on edge devices.

However, existing LUT-based SR methods still have some drawbacks. When creating LUTs, all inputs need to be traversed, which makes the size of a single LUT grow exponentially with the increase of input pixels. Considering the limitation of memory size, the number of input pixels must be controlled to reduce the LUT size, further limiting the receptive field size. Therefore, the performance of LUT-based SR methods is usually limited. Although recent studies have enlarged the receptive field by using multiple LUTs (Ma et al. 2022; Li et al. 2022), the total size of the corresponding LUTs grows linearly with their indexing capability, thus improving SR performance while ensuring practicality. Nevertheless, using multiple LUTs for indexing increases the memory access time and makes these methods lose their advantage in inference speed.

To address these issues, in this paper, we propose an efficient expanded convolution (EC) method that indirectly enlarges the RF by expanding the output size of the convolution. This operation helps to reduce the LUT size comparing to increasing the size from the input end and is more friendly for the LUT query operation. Therefore, the proposed method has a great advantage in inference speed. Specifically, we follow the SR-LUT approach overall, but with two differences. On the one hand, we add an EC layer at the end of the network. The output of the EC contains the output at the target position and the outputs around the target position. Compared with expanding the input, expanding the output also achieves the effect of enlarging the RF, and makes the LUT grow linearly. Moreover, since only a single query is required in the inference phase, our method is faster than using multiple LUTs. On the other hand, since discretizing the output with 8bit storage introduces errors, we expand the numerical range of the sum of index results and rescale the result to [0, 255] to obtain the final result, which effectively improves the error introduced by quantization. Since there is no floating-point operation in the whole inference process and only a single LUT is used, our method has very fast inference speed. More importantly, our method has great flexibility and can be easily combined with other methods. For example, using multiple LUTs to sacrifice more inference time for better inference quality. In summary, our contributions can be summarized as follows:

• We propose a novel expanded convolution (EC) method that enlarges the RF by expanding the output size of the convolution. The EC method is very friendly for the LUT, as it can merge multiple look-ups into one and give the LUT more advantage in speed of inference.
• We propose a method that is simple and effective to mitigate the error caused by quantizing the result to 8 bits during the LUT recording process by using a scaling factor to multiply the accumulated result. The computational cost of this method is negligible.
• Experimental results show that our method has a significant improvement in inference speed compared with other LUT-based SR methods, while having comparable performance, demonstrating its practicality.

Related Work

Single Image Super-Resolution

Deep neural networks have powerful fitting capabilities. With the development of deep learning, models based on deep learning have made significant progress in SISR tasks. Dong et al. (Dong et al. 2014) first successfully attempted to use a three-layer convolutional neural network for super-resolution, SRCNN, which is the groundbreaking work of SR based on deep learning. SRCNN requires performing bicubic interpolation on the image to up-sampling to the target size before using CNNs to refine the results. In contrast, FSRCNN (Dong, Loy, and Tang 2016) eliminates upsampling at the input end and up-scales at the output end, improving computational efficiency. Shi et al. (Shi et al. 2016) proposed the ESPCNN, which introduced a sub-pixel convolutional layer and achieved real-time super-resolution by zooming in on images at the end of the network. Ledig et al. (Ledig et al. 2017) introduced adversarial learning, which can add more details to images, making them more natural and realistic.

With the further development of deep learning, it has become a consensus that deeper networks have stronger feature representation and fitting capabilities. By increasing depth and using residual connections, VDSR (Kim, Lee, and Lee 2016a) improves performance. Lim et al. (Lim et al. 2017) further increased the depth and width of the network and proposed a model called EDSR. Zhang et al. (Zhang et al. 2018a) further improved SR performance by introducing channel attention. Kim et al. (Kim, Lee, and Lee 2016b) first used recursive methods to increase network depth, known as DRCN. Tai et al. (Tai, Yang, and Liu 2017) further proposed DRRN by incorporating local residual learning into DRCN. Ahn et al. (Ahn, Kang, and Sohn 2018) proposed a cascaded residual network, which integrates features from different layers using a cascaded framework at both local and global levels, achieving high SR performance with fewer parameters. Later, Ledig et al. (Ledig et al. 2017) proposed SRResNet, which uses dense connections for improvement. These methods using residual learning have achieved significant improvements compared to traditional SR methods, demonstrating the effectiveness of residual networks.

In addition to improving super-resolution performance, reducing model parameters and accelerating running speed is another worthwhile direction. Using a pyramid framework, the LapSRN proposed by Lai et al. (Lai et al. 2018) can effectively perform SR at extremely low resolutions.
Later, Muqeet et al. (Muqeet et al. 2020) proposed stacked multi-attention blocks, effectively compensating for parameter loss. However, these methods use very deep networks, consuming a large amount of computing resources during execution.

**Look-up Table**

Space-time tradeoff is a common strategy, and the look-up table is a space-time tradeoff method. The look-up table records the output results corresponding to all inputs in advance and uses simple and fast memory access operations instead of complex and lengthy computations, thus significantly improving the algorithm running efficiency. LUT has many applications, such as color space conversion (Monga et al. 2016), numerical computation (Rizvi et al. 1995; Chin-Chen et al. 2000), and video coding (Lee, Lee, and Park 2010; Tsang, Chan, and Siu 2013). In addition, LUT is also commonly used in camera imaging pipelines (Karaimer and Brown 2016). After the rise of deep learning, many studies have proposed LUT methods combined with deep learning for low-level vision tasks (Wang et al. 2021a,b; Zeng et al. 2022). Zeng et al. (Zeng et al. 2022) first proposed an adaptive 3D look-up table for image enhancement, and Wang et al. (Wang et al. 2021b) further proposed a learnable spatially aware 3D look-up table. Jo et al. (Jo and Kim 2021) first proposed a LUT-based SR method by training an SR network and then transferring the mapping relationship in the SR network to LUT. Combined with Rotational Ensemble and interpolation techniques, they developed SR-LUT. Li et al. (Li et al. 2022) designed a new indexing scheme based on SR-LUT by using multiple LUTs to enlarge the receptive field, and they improved the SR performance. Ma et al. (Ma et al. 2022) proposed SP-LUT, which also used multiple cascaded LUTs to enlarge the receptive field by using MSB and LSB, two parallel branches to compensate for the accuracy loss caused by discretization. They avoided interpolation operations.

**Method**

**Preliminary**

Given an LR image \(I^{LR}\), the goal of SR task is to generate an HR image \(I^{SR}\) that is close to the ground-truth image \(I^{HR}\). For each pixel \((i, j)\) on the SR image, we think that it is mapped from the pixels around \((i', j')\) on the input image. Let \(\Phi\) be the mapping function for an SR network. The input of \(\Phi\) is all the pixels covered by the RF of the SR network. LUT-based SR methods first find an \(\Phi\), then traverse all possible inputs of \(\Phi\) to generate corresponding outputs. These input data and corresponding output data form key-value pairs. We record these data in LUT, where the key is implicit in the index, and the output can be obtained by querying LUT with the input as the index. The inference process is to read the pixels covered by RF at position \((i', j')\) on LR in turn, combine them as an index, retrieve the result from LUT, and write it to the corresponding position \((i, j)\) on SR.

Generally speaking, the larger the RF, the stronger the mapping ability of \(\Phi\), and the better the SR performance.
However, as mentioned before, LUT records all input-output pairs. As RF expands, the domain of $\Phi$ expands exponentially, which means that the size of LUT will grow dramatically. As discussed by (Jo and Kim 2021), suppose LUT stores 8bit input-output, magnification factor $r = 4$, when RF covers two pixels, the size of LUT is 1MB. When RF covers 3, 4, and 5 pixels, the size of LUT is 256MB, 64GB, and 16TB, respectively. The large size of LUT not only consumes more storage space but also may increase the time of retrieving LUT. Therefore, in order to control the size of LUT, some tricks are needed, such as rotational ensemble, processing each channel separately, sampling LUT, etc.

**Expand LUT**

Convolution operation can map an input of size $k_s \times k_s$ to an output of size $1 \times 1$. As pointed out by Dong et al. (Dong et al. 2014), there is no efficient implementation of convolutional layers with an output size larger than the input size. Subsequently, Shi et al. (Shi et al. 2016) proposed sub-pixel convolution, which achieves the magnification operation of SR by rearranging the elements in the tensor. Inspired by this, we propose expanded convolution, which also achieves the magnification operation at the output end of the convolution by rearranging the elements in the tensor. Compared with the original sub-pixel convolution, we go further and enlarge the output size of a single convolution operation to $r \times k \times r \times k$. As the convolution kernel slides on the input, a sliding window with overlap is formed at the output end. The output of the convolution is obtained by summing up the overlapping parts. This process can be expressed as follows:

$$X(i, j) = \Phi_0(F(i, j)), \quad I_{SR}(i, j) = \sum_{x \in \chi} x(i, j)$$

where $X(i, j)$ denotes the sliding window obtained by the convolution operation at the position $(i, j)$. $F(i, j)$ denotes the feature mapping function, i.e. the expanded convolution. $F(i, j)$ is the input feature at the position $(i, j)$. $I(i, j)$ represents the output of the EC, and $\chi$ denotes the sliding window sets that cover the position $(i, j)$. In practice, we do not actually redesign the operator but achieve the goal by padding the tensor. Figure 3 shows the specific implementation process. We output $r \times r \times 9$ channels at the last layer of the network and then rearrange the tensor to obtain a tensor of shape $9 \times rH \times rW$. Next, we pad the separated nine tensors in different directions so that their RFs will shift in different directions. After adding up the nine feature maps, the
resulting feature map has a larger RF. Note that this process is similar to the working principle of the aggregation module of SP-LUT. The only difference is that SP-LUT uses horizontal aggregation module and vertical aggregation module separately, while our method generalizes to any case.

Figure 4 visualizes the change process of the RF. During inference, two pixels are read from the input image, then the LUT is queried, and the retrieved data is placed on the output matrix according to the position and added in place. After one query, the RF changes from $1 \times 2$ to $3 \times 4$. Further, using the rotation trick, the RF can be enlarged to 21 pixels. Since the above operations can be completed at the position $(i, j)$ simultaneously, our method has a faster inference speed.

**Rescale Output**

When training an SR network, 32-bit or 64-bit floating-point numbers are usually used, which have high precision. During inference, the LUT replaces the SR network to perform the mapping. When we transfer the mapping relationship of SR network to LUT, in order to further reduce the LUT size, we adjust the output to 8bit, with a numerical range of $[-128, 127]$. If LUT stores the final output result, then 8bit just matches the single-channel bit width of a general image. However, since we use the rotational ensemble and expanded output methods, LUT actually stores only an intermediate result, and the final result still needs to go through multiple addition operations. The inference process can be expressed as follows:

$$X(i, j) = LUT[I^{LR}(it, jt)]$$

where $I^{LR}(it, jt)$ denotes pixels on the input image used for querying. Quantizing the floating-point number in $[0, 1]$ to the integer in $[-128, 127]$ itself will cause some errors, and these errors will accumulate after multiple addition operations, eventually affecting the SR performance. Therefore, we propose to expand the numerical range of the accumulation results, and readjust them back to the original numerical range after the accumulation. This process can be expressed as:

$$I^{SR}(i, j) = \text{clamp}(\text{round}(\alpha \cdot \sum_{x \in X} x(i, j)))$$

where $\text{clamp}$ denotes clipping the result to $[0, 255]$, $\text{round}$ denotes the rounding function, and $\alpha$ is the scaling factor. By expanding the numerical range and multiplying by a scaling factor (in practice, we take $\alpha$ as 0.25), the numerical range is pulled back to $[0, 255]$. This operation can be seen as adjusting the gradation value of $x(i, j)$ from 1 to 0.25. Experimental results show that this operation effectively improves the error. At the same time, the operation of multiplying by 0.25 and then rounding can be implemented by adding 2 and then shifting right by two bits, which has a very small computational cost.

**Experiment**

**Implementation Details**

**Datasets and Metrics.** Following previous studies, we use the DIV2K dataset (Timofte et al. 2017) for training. This dataset has 800 images for training, 100 images for validation, and 100 images for testing. In addition, there are five commonly used benchmark test datasets, namely Set5 (Bevilacqua et al. 2012), Set14 (Zeyde, Elad, and Protter 2012), B100 (Arbelaez et al. 2010), Urban100 (Huang, Singh, and Ahuja 2015) and Mang109 (Matsui et al. 2017). We report our results on these five datasets and compare them with previous studies. The quantitative evaluation metrics are PSNR (peak signal-to-noise ratio) on the Y channel of YCbCr space and structural similarity index (SSIM) (Wang et al. 2004). In addition, we evaluate the computation efficiency by recording and presenting the runtime of generating $1280 \times 720$ output images on mobile devices. To be consistent with previous studies, according to (Lim et al. 2017; Zhang et al. 2018b), we use Matlab’s imresize function to perform bicubic interpolation on HR images to obtain LR images.

**Training Details.** Our convolutional network structure is basically consistent with SR-LUT (Jo and Kim 2021), with the first convolutional layer having a kernel size of $1 \times 2$ and the rest having a kernel size of $1 \times 1$. The difference is that we use 128 channels of convolution and add expanded convolution, and also scale the output. We use Adam optimizer (Kingma and Ba 2015) with an initial learning rate of $1 \times 10^{-4}$ for a total of 20000 epochs, halving the learning rate every 4000 epochs. The loss function is mean-squared error (MSE). We randomly crop the LR image into patches of size $48 \times 48$ with a mini-batch size of 16 and augment the data by random rotation and flipping. We train the EC-LUT model with Pytorch (Chaudhary et al. 2020) on Nvidia 2080Ti GPU.
Table 1: Quantitative comparisons with other SR methods on 5 benchmark datasets for $r = 4$. The best values of LUT-based methods are shown in bold. Runtime is measured on a MEIZU 16s smartphone for generating $1280 \times 720$ output images. Size denotes the storage space or the parameter number of each model.

<table>
<thead>
<tr>
<th>Method</th>
<th>Runtime</th>
<th>Size</th>
<th>Set5</th>
<th>Set14</th>
<th>BSDS100</th>
<th>Urban100</th>
<th>Manga109</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest</td>
<td>9ms</td>
<td>-</td>
<td>26.25/0.7372</td>
<td>24.65/0.6529</td>
<td>25.03/0.6293</td>
<td>22.17/0.6154</td>
<td>23.45/0.7414</td>
</tr>
<tr>
<td>Bilinear</td>
<td>20ms</td>
<td>-</td>
<td>27.55/0.7884</td>
<td>25.42/0.6792</td>
<td>25.54/0.6460</td>
<td>22.69/0.6346</td>
<td>24.21/0.7666</td>
</tr>
<tr>
<td>Bicubic</td>
<td>97ms</td>
<td>-</td>
<td>28.42/0.8101</td>
<td>26.00/0.7023</td>
<td>25.96/0.6672</td>
<td>23.14/0.6574</td>
<td>24.91/0.7871</td>
</tr>
<tr>
<td>SR-LUT</td>
<td>94ms</td>
<td>1.274MB</td>
<td>29.82/0.8478</td>
<td>27.01/0.7355</td>
<td>26.53/0.6953</td>
<td>24.02/0.6990</td>
<td>26.80/0.8380</td>
</tr>
<tr>
<td>SP-LUT</td>
<td>365ms</td>
<td>5.5MB</td>
<td>30.01/0.8516</td>
<td>27.21/0.7427</td>
<td>26.67/0.7019</td>
<td>24.12/0.7058</td>
<td>27.00/0.8430</td>
</tr>
<tr>
<td>Mu-LUT</td>
<td>242ms</td>
<td>4.062MB</td>
<td>30.60/0.8653</td>
<td>27.60/0.7541</td>
<td>26.86/0.7110</td>
<td>24.46/0.7194</td>
<td>27.90/0.8633</td>
</tr>
<tr>
<td>Ours</td>
<td>41ms</td>
<td>9MB</td>
<td>29.91/0.8461</td>
<td>27.14/0.7419</td>
<td>26.61/0.7019</td>
<td>23.98/0.6977</td>
<td>26.96/0.8362</td>
</tr>
<tr>
<td>FSR-CNN</td>
<td>371ms</td>
<td>12K</td>
<td>30.71/0.8656</td>
<td>27.60/0.7543</td>
<td>26.96/0.7129</td>
<td>24.61/0.7263</td>
<td>27.91/0.8587</td>
</tr>
<tr>
<td>CARN-M</td>
<td>4955ms</td>
<td>412K</td>
<td>31.82/0.8998</td>
<td>2829/0.7747</td>
<td>27.42/0.7305</td>
<td>25.62/0.7694</td>
<td>29.85/0.8993</td>
</tr>
<tr>
<td>RRDB</td>
<td>31717ms</td>
<td>16698K</td>
<td>32.68/0.8999</td>
<td>28.88/0.7891</td>
<td>27.82/0.7444</td>
<td>27.02/0.8146</td>
<td>31.57/0.9185</td>
</tr>
</tbody>
</table>

Figure 5: Visual comparison for $\times 4$ SR on benchmark datasets. The results show our method can generate comparable results compared to other LUT-based methods.

Comparaison with Others

Quantitative Comparison. In this section, we compare our proposed method with three common interpolation methods, including nearest neighbor, bilinear, bicubic (Keys 1981) interpolation, LUT-based SR methods including SR-LUT (Jo and Kim 2021), MuLUT (Li et al. 2022), SP-LUT (Ma et al. 2022), and DNN-based methods including FSR-CNN (Dong, Loy, and Tang 2016), CARN-M (Ahn, Kang, and Sohn 2018), RRDB (Wang et al. 2018). Since MuLUT did not provide source code, we used the numbers reported in their paper. To be consistent with previous studies, we perform four times SR on input images of size 320x180, and measure the running time.

A quantitative comparison is shown in Table 1. As we can see, EC-LUT has better inference time than any other methods, except for nearest neighbor and bilinear interpolation. Compared with the three interpolation methods, EC-LUT has better PSNR and SSIM (on Set5 test set, +1.49dB and +0.036, respectively, compared with bicubic interpolation). Although DNN-based methods generally have better PSNR and SSIM than LUT-based methods, their inference time is tens to hundreds of times longer than EC-LUT. In compari-
Table 2: Ablation study of rescale operation, where no quant means that the output of the SR network is not quantized and 32bit floating-point numbers are used directly. We use bit shift operations to replace multiplication. The results show that the SR performance degrades after quantization, but it recovers by shifting the accumulated result two bits to the right ($\alpha = 0.25$).

<table>
<thead>
<tr>
<th>Right Shift</th>
<th>Se5 PSNR</th>
<th>Se5 SSIM</th>
<th>Set14 PSNR</th>
<th>Set14 SSIM</th>
<th>BSDS100 PSNR</th>
<th>BSDS100 SSIM</th>
<th>Urban100 PSNR</th>
<th>Urban100 SSIM</th>
<th>Manga109 PSNR</th>
<th>Manga109 SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 bit</td>
<td>29.91</td>
<td>0.8454</td>
<td>27.14</td>
<td>0.7419</td>
<td>26.61</td>
<td>0.7019</td>
<td>23.98</td>
<td>0.6977</td>
<td>26.96</td>
<td>0.8362</td>
</tr>
<tr>
<td>1 bit</td>
<td>29.91</td>
<td>0.8461</td>
<td>27.14</td>
<td>0.7419</td>
<td>26.61</td>
<td>0.7019</td>
<td>23.98</td>
<td>0.6977</td>
<td>26.96</td>
<td>0.8362</td>
</tr>
<tr>
<td>2 bit</td>
<td>29.91</td>
<td>0.8461</td>
<td>27.14</td>
<td>0.7419</td>
<td>26.61</td>
<td>0.7019</td>
<td>23.98</td>
<td>0.6977</td>
<td>26.96</td>
<td>0.8362</td>
</tr>
<tr>
<td>3 bit</td>
<td>29.91</td>
<td>0.8459</td>
<td>27.08</td>
<td>0.7408</td>
<td>26.57</td>
<td>0.7006</td>
<td>23.93</td>
<td>0.6966</td>
<td>26.78</td>
<td>0.8341</td>
</tr>
<tr>
<td>no quant</td>
<td>29.91</td>
<td>0.8470</td>
<td>27.13</td>
<td>0.7414</td>
<td>26.60</td>
<td>0.7015</td>
<td>23.96</td>
<td>0.6975</td>
<td>26.88</td>
<td>0.8344</td>
</tr>
</tbody>
</table>

Table 3: Experiments on different kernel shapes. The kernel shape of V, F and S models are $1 \times 2$, $1 \times 3$ and $2 \times 2$ respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>Se5 PSNR</th>
<th>Se5 SSIM</th>
<th>Set14 PSNR</th>
<th>Set14 SSIM</th>
<th>BSDS100 PSNR</th>
<th>BSDS100 SSIM</th>
<th>Urban100 PSNR</th>
<th>Urban100 SSIM</th>
<th>Manga109 PSNR</th>
<th>Manga109 SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC-LUT-V</td>
<td>29.91</td>
<td>0.8461</td>
<td>27.14</td>
<td>0.7419</td>
<td>26.61</td>
<td>0.7019</td>
<td>23.98</td>
<td>0.6977</td>
<td>26.96</td>
<td>0.8362</td>
</tr>
<tr>
<td>EC-LUT-F</td>
<td>30.24</td>
<td>0.8555</td>
<td>27.39</td>
<td>0.7488</td>
<td>26.76</td>
<td>0.7078</td>
<td>24.18</td>
<td>0.7082</td>
<td>27.41</td>
<td>0.8487</td>
</tr>
<tr>
<td>EC-LUT-S</td>
<td>30.35</td>
<td>0.8592</td>
<td>27.45</td>
<td>0.7484</td>
<td>26.77</td>
<td>0.7062</td>
<td>24.28</td>
<td>0.7101</td>
<td>27.39</td>
<td>0.8466</td>
</tr>
</tbody>
</table>

Conclusion

In this paper, we propose a novel LUT-based single image SR method, which enlarges the RF effect by expanding the coverage range at the output end and improves the SR performance by using only one LUT. At the same time, as only one single LUT is used, our method has an absolute advantage in inference speed. On the other hand, we propose a method to mitigate quantization error by rescaling the accumulated results at the output end, which greatly reduces the SR performance degradation caused by quantization. Compared with previous studies, our method has surpassed or comparable performance while greatly improving the inference speed, further enhancing the practicality of the method. In the future, we will explore the application of the EC method in multiple LUTs to further improve SR performance.
References


