RankSRGAN: Generative Adversarial Networks with Ranker
for Image Super-Resolution

Wenlong Zhang1  Yihao Liu1,2  Chao Dong1,†  Yu Qiao1
1ShenZhen Key Lab of Computer Vision and Pattern Recognition, SIAT-SenseTime Joint Lab, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
2University of Chinese Academy of Sciences
{wl.zhang1, yh.liu4, chao.dong, yu.qiao}@siat.ac.cn

Abstract

Generative Adversarial Networks (GAN) have demonstrated the potential to recover realistic details for single image super-resolution (SISR). To further improve the visual quality of super-resolved results, PIRM2018-SR Challenge employed perceptual metrics to assess the perceptual quality, such as PI, NIQE, and Ma. However, existing methods cannot directly optimize these indifferentiable perceptual metrics, which are shown to be highly correlated with human ratings. To address the problem, we propose Super-Resolution Generative Adversarial Networks with Ranker (RankSRGAN) to optimize generator in the direction of perceptual metrics. Specifically, we first train a Ranker which can learn the behavior of perceptual metrics and then introduce a novel rank-content loss to optimize the perceptual quality. The most appealing part is that the proposed method can combine the strengths of different SR methods to generate better results. Extensive experiments show that RankSRGAN achieves visually pleasing results and reaches state-of-the-art performance in perceptual metrics. Project page: https://wenlongzhang0724.github.io/Projects/RankSRGAN

1. Introduction

Single image super resolution aims at reconstructing/generating a high-resolution (HR) image from a low-resolution (LR) observation. Thanks to the strong learning capability, Convolutional Neural Networks (CNNs) have demonstrated superior performance [10, 24, 42] to the conventional example-based [40] and interpolation-based [41] algorithms. Recent CNN-based methods can be divided into two groups. The first one regards SR as a reconstruction problem and adopts MSE as the loss function to achieve high PSNR values. However, due to the conflict between the reconstruction accuracy and visual quality, they tend to produce overly smoothed/sharpened images. To favor better visual quality, the second group casts SR as an image generation problem [22]. By incorporating the perceptual loss [6, 18] and adversarial learning [22], these perceptual SR methods have potential to generate realistic textures and details, thus attracted increasing attention in recent years.

The most challenging problem faced with perceptual SR methods is the evaluation. Most related works resort to user study for subjectively evaluating the visual quality [2, 34]. However, without an objective metric like PSNR/SSIM, it is hard to compare different algorithms on a fair platform, which largely prevents them from rapid development. To address this issue, a number of no-reference image quality assessment (NR-IQA) metrics are proposed, and some of them are proven to be highly correlated with human ratings [2], such as NIQE [29] (correlation 0.76) and PI [2] (correlation 0.83). Specially, the PIRM2018-SR challenge [2] introduced the PI metric as perceptual criteria and successfully ranked the entries. Nevertheless, most of these NR-IQA metrics are not differentiable (e.g., they include hand-crafted feature extraction or statistic regression operation), making them infeasible to serve as loss functions.

1Corresponding author (e-mail: chao.dong@siat.ac.cn)
Considering NR-IQA metrics in optimization, existing perceptual SR methods could not show stable performance in the orientation of objective perceptual criteria.

To overcome this obstacle, we propose a general and differentiable model – Ranker, which can mimic any NR-IQA metric and provide a clear goal (as loss function) for optimizing perceptual quality. Specifically, Ranker is a Siamese CNN that simulates the behavior of the perceptual metric by learning to rank approach [7]. Notably, as NR-IQA metrics have various dynamic ranges, Ranker learns their output ranking orders instead of absolute values. Just like in the real world, people tend to rank the quality of images rather than give a specific value. We equip Ranker with the standard SRGAN model and form a new perceptual SR framework – RankSRGAN (Super-Resolution Generative Adversarial Networks with Ranker). In addition to SRGAN, the proposed framework has a rank-content loss using a well-trained Ranker to measure the output image quality. Then the SR model can be stably optimized in the orientation of specific perceptual metrics.

To train the proposed Ranker, we prepare another training dataset by labeling the outputs of different SR algorithms. Then the Ranker, with a Siamese-like architecture, could learn these ranking orders with high accuracy. The effectiveness of the Ranker is largely determined by the selected SR algorithms. To achieve the best performance, we adopt two state-of-the-art perceptual SR models – SRGAN [22] and ESRGAN [35]. As the champion of PIRM2018-SR challenge [2], ESRGAN is superior to SRGAN on average scores, but can not outperform SRGAN on all test images. When evaluating with NIQE [29], we obtain mixed orders for these two methods. Then the Ranker will favor different algorithms on different images, rather than simply classifying an image into a binary class (SRGAN/ESRGAN). After adopting the rank-content loss, the generative network will output results with higher ranking scores. In other words, the learned SR model could combine the better parts of SRGAN and ESRGAN, and achieve superior performance both in perceptual metric and visual quality. Figure 1 shows an example of RankSRGAN, which fuses the imagery effects of SRGAN and ESRGAN and obtains better NIQE score.

We have done comprehensive ablation studies to further validate the effectiveness of the proposed method. First, we distinguish our Ranker from the regression/classification network that could also mimic the perceptual metric. Then, we train and test RankSRGAN with several perceptual metrics (i.e. NIQE [29], Ma [26], PI [2]). We further show that adopting different SR algorithms to build the dataset achieves different performance. Besides, we have also investigated the effect of different loss designs and combinations. With proper formulation, our method can clearly surpass ESRGAN and achieve state-of-the-art performance.

In summary, the contributions of this paper are threefold. (1) We propose a general perceptual SR framework – RankSRGAN that can optimize generator in the direction of indistinguishable perceptual metrics and achieve the state-of-the-art performance. (2) We, for the first time, utilize results of other SR methods to build training dataset. The proposed method combines the strengths of different SR methods and generates better results. (3) The proposed SR framework is highly flexible and produce diverse results given different rank datasets, perceptual metrics, and loss combinations.

2. Related work

Super resolution. Since Dong et al. [10] first introduced convolutional neural networks (CNNs) to the SR task, a series of learning-based works [40, 15, 20, 16, 13, 14] have achieved great improvements in terms of PSNR. For example, Kim et al. [20] propose a deep network VDSR with gradient clipping. The residual and dense block [24, 42] are explored to improve the super-resolved results. In addition, SRGAN [22] is proposed to generate more realistic images. Then, texture matching [31] and semantic prior [34] are introduced to improve perceptual quality. Furthermore, the perceptual index [2] consisting of NIQE [29] and Ma [26] is adopted to measure the perceptual SR methods in the PIRM2018-SR Challenge at ECCV [2]. In the Challenge, ESRGAN [35] achieves the state-of-the-art performance by improving network architecture and loss functions.

CNN for NR-IQA. No-reference Image Quality Assessment (NR-IQA) can be implemented by learning-based models, which extract hand-crafted features from Natural Scene Statistics (NSS), such as CBIQ [37], NIQE [29], and Ma [26], etc. In [23], Li et al. develop a general regression neural network to fit human subjective opinion scores with pre-extracted features. Kang et al. [19, 4] integrate a general CNN framework which can predict image quality on local regions. In addition, Liu et al. [25] propose RankIQA to tackle the problem of lacking human-annotated data in NR-IQA. They first generate large distorted images in different distortion level. Then they train a Siamese Network to learn the rank of the quality of those images, which can improve the performance of the image quality scores.

Learning to rank. It has been demonstrated that learning to rank approach is effective in computer vision. For instance, Devi Parikh et al. [30] model relative attributes using a well-learned ranking function. Yang et al. [36] first employ CNN for relative attribute ranking in a unified framework. One of the most relevant studies to our work is RankCGAN [32], which investigates the use of GAN to tackle the task of image generation with semantic attributes. Unlike standard GANs that generate the image from noise input (CGAN [28]), RankCGAN incorporates a pairwise Ranker into CGAN architecture so that it can handle continuous attribute values with subjective measures.
3. Method

3.1. Overview of RankSRGAN

The proposed framework is built upon the GAN-based [22] SR approach, which consists of a generator and a discriminator. The discriminator network tries to distinguish the ground-truth images from the super-resolved results, while the generator network is trained to fool the discriminator. To obtain more natural textures, we propose to add additional constraints on the standard SRGAN [22] by exploiting the prior knowledge of perceptual metrics to improve the visual quality of output images. The overall framework of our approach is depicted in Figure 2. The pipeline involves the following three stages:

Stage 1: Generate pair-wise rank images. First, we employ different SR methods to generate super-resolved images on public SR datasets. Then we apply a chosen perceptual metric (e.g. NIQE) on the generated images. After that, we can pick up two images of the same content to form a pair and rank the pair-wise images according to the quality score calculated by the perceptual metric. Finally, we obtain the pair-wise images and the associated ranking labels. More details will be presented in Section 4.1.

Stage 2: Train Ranker. The Ranker adopts a Siamese architecture to learn the behavior of perceptual metrics and the network structure is depicted in Section 3.2. We adopt margin-ranking loss, which is commonly used in “learning to rank” [7], as the cost function to optimize Ranker. The learned Ranker is supposed to have the ability to rank images according to their perceptual scores.

Stage 3: Introduce rank-content loss. Once the Ranker is well-trained, we use it to define a rank-content loss for a standard SRGAN to generate visually pleasing images. Please see the rank-content loss in Section 3.3.

3.2. Ranker

Rank dataset. Similar to [8, 25], we use super-resolution results of different SR methods to represent different perceptual levels. With a given perceptual metric, we can rank these results in a pair-wise manner. Picking any two SR images, we can get their ranking order according to the quality score measured by the perceptual metric. These pair-wise data with ranking labels form a new dataset, which is defined as the rank dataset. Then we let the proposed Ranker learn the ranking orders. Specifically, given two input images $y_1$, $y_2$, the ranking scores $s_1$ and $s_2$ can be obtained by

$$s_1 = R(y_1; \Theta_R)$$

$$s_2 = R(y_2; \Theta_R),$$

where $\Theta_R$ represents the network weights and $R(\cdot)$ indicates the mapping function of Ranker. In order to make the Ranker output similar ranking orders as the perceptual metric, we can formulate:

$$\begin{cases} s_1 < s_2 & \text{if } m_{y_1} < m_{y_2} \\ s_1 > s_2 & \text{if } m_{y_1} > m_{y_2} \end{cases}$$

where $m_{y_1}$ and $m_{y_2}$ represent the quality scores of image $y_1$ and image $y_2$, respectively. A well-trained Ranker could guide the SR model to be optimized in the orientation of the given perceptual metric.

Siamese architecture. The Ranker uses a Siamese-like architecture [5, 9, 38], which is effective for pair-wise inputs. The architecture of Ranker is shown in Figure 2. It
has two identical network branches which contain a series of convolutional, LeakyReLU, pooling and full-connected layers. Here we use a Global Average Pooling layer after the Feature Extractor, thus the architecture can get rid of the limit of input size. To obtain the ranking scores, we employ a fully-connected layer as a regressor to quantify the rank results. Note that we do not aim to predict the real values of the perceptual metric since we only care about the ranking information. Finally, the outputs of two branches are passed to the margin-ranking loss module, where we can compute the gradients and apply back-propagation to update parameters of the whole network.

Optimization. To train Ranker, we employ margin-ranking loss that is commonly used in sorting problems [36, 25]. The margin-ranking loss is given below:

\[
L(s_1, s_2; \gamma) = \max(0, (s_1 - s_2) * \gamma + \varepsilon)
\]

where \( s_1 \) and \( s_2 \) represent the ranking scores of pair-wise images. The \( \gamma \) is the rank label of the pair-wise training images. The margin \( \varepsilon \) can control the distance between \( s_1 \) and \( s_2 \). Therefore, the \( N \) pair-wise training images can be optimized by:

\[
\hat{\Theta} = \arg \min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} L(s_1^{(i)}, s_2^{(i)}; \gamma^{(i)})
\]

3.3. RankSRGAN

RankSRGAN consists of a standard SRGAN and the proposed Ranker, as shown in Figure 2. Compared with existing SRGAN, our framework simply adds a well-trained Ranker to constrain the generator in SR space. To obtain visually pleasing super-resolved results, adversarial learning [22, 31] is applied to our framework where the generator and discriminator are jointly optimized with the objective

\[
\min \max \mathbb{E}_{y \sim p_{HL}} \log D_H(y) + \mathbb{E}_{y \sim p_{LR}} \log(1 - D_S(G_{\Theta}(x))),
\]

where \( p_{HR} \) and \( p_{LR} \) represent the probability distributions of \( HR \) and \( LR \) samples, respectively. In order to demonstrate the effectiveness of the proposed Ranker, we do not use complex architectural designs of GAN [35] but use the general SRGAN [22].

Perceptual loss. In [12, 18], the perceptual loss is proposed to measure the perceptual similarity between two images. Instead of computing distances in image pixel space, the images are first mapped into feature space and the perceptual loss can be presented as:

\[
L_P = \sum_i ||\phi(y_i) - \phi(y_i)||_2^2,
\]

where \( \phi(y_i) \) and \( \phi(y_i) \) represent the feature maps of HR and SR images, respectively. Here \( \phi \) is obtained by the 5-th convolution (before maxpooling) layer within VGG19 network [33].

Adversarial loss. Adversarial training [22, 31] is recently used to produce natural-looking images. A discriminator is trained to distinguish the real image from the generated image. This is a minimax game approach where the generator loss \( L_G \) is defined based on the output of discriminator:

\[
L_G = -\log D(G(x)),
\]

where \( x \) is the LR image, \( D(G(x)) \) represents the probability of the discriminator over all training samples.

Rank-content loss. The generated image is put into the Ranker to predict the ranking score. Then, the rank-content loss can be defined as:

\[
L_R = \text{sigmoid}(R(G(x))),
\]

where \( R(G(x)) \) is the ranking score of the generated image. A lower ranking score indicates better perceptual quality. After applying the sigmoid function, \( L_R \) represents ranking-content loss ranging from 0 to 1.

3.4. Analysis of Ranker

The proposed Ranker possesses an appealing property: by elaborately selecting the SR algorithms and the perceptual metric, the RankSRGAN has the potential to surpass the upper bound of these methods and achieve superior performance. To validate this comment, we select the state-of-the-art perceptual SR methods – SRGAN [22] and ESRGAN [35] to build the rank dataset. Then we use the perceptual metric NIQE [29] for evaluation. NIQE is demonstrated to be highly correlated with human ratings and easy to implement. A lower NIQE value indicates better perceptual quality. When measured with NIQE on the PIRM-Test [2] dataset, the average scores of SRGAN and ESRGAN are 2.70 and 2.55, respectively. ESRGAN obtains better NIQE scores for most images but not all images, indicating that SRGAN and ESRGAN have mixed ranking orders with NIQE.

In order to examine the effectiveness of our proposed Ranker, we compare two ranking strategies – metric rank and model classification. Metric rank, which is our proposed method, uses perceptual metrics to rank the images. For example, in each image pair, the one with a lower score is labeled to 1 and the other is 2. The model classification, as the comparison method, ranks images according to the used SR methods, i.e., all results of ESRGAN are labeled to 1 and those of SRGAN are labeled to 2. We then give an analysis of the upper bound of these two methods. The upper bound can be calculated as:

\[
UB_{MC} = \text{Mean}(PM_{SR-2-L} + PM_{SR-2-H})
\]

\[
UB_{MR} = \text{Mean}(PM_{SR-2-L} + PM_{SR-1-L})
\]

where \( PM_{SR-1-L} < PM_{SR-2-H} \), where \( UB_{MC} \) and \( UB_{MR} \) represent the upper bound of model classification and metric rank, respectively. PM
Model Classification
5.968 2.705 2.557
ESRGAN 28.33 25.62 25.30
SRResNet SRGAN ESRGAN

eels, as shown in Table 1. To generate super-resolved images as three perceptual levels (SRResNet [22], SRGAN [22] and ESRGAN [35]) are used to train separate RankSRGAN models. As shown in Figure 3, Obviously, metric rank could combine the better parts of different algorithms and exceed the upper bound of a single algorithm.

We further conduct SR experiments to support the above analysis. We use the metric rank and model classification approach to label the rank dataset. Then the Ranker-MC (model classification) and Ranker-MR (metric rank) are used to train separate RankSRGAN models. Figure 4 shows the quantitative results (NIQE), where RankSRGAN-MR outperforms ESRGAN and RankSRGAN-MC. This demonstrates that our method can exceed the upper bound of all chosen SR algorithms.

4. Experiments

4.1. Training details of Ranker

Datasets. We use DIV2K (800 images) [1] and Flickr2K (2650 images) [1] datasets to generate pair-wise images as rank dataset for training. Three different SR algorithms (SRResNet [22], SRGAN [22] and ESRGAN [35]) are used to generate super-resolved images as three perceptual levels, as shown in Table 1.

<table>
<thead>
<tr>
<th>PIRM-Test</th>
<th>SRResNet</th>
<th>SRGAN</th>
<th>ESRGAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIQE</td>
<td>5.968</td>
<td>2.705</td>
<td>2.557</td>
</tr>
<tr>
<td>PSNR</td>
<td>28.33</td>
<td>25.62</td>
<td>25.30</td>
</tr>
</tbody>
</table>

Table 1. The performance of super-resolved results with three perceptual levels in PIRM-Test [2].

We extract patches from those pair-wise images with a stride of 200 and size of 296 × 296. For one perceptual level (SR algorithm), we can generate 150 K patches (10% for validation, 90% for training). Inspired by PIRM2018-SR Challenge [2], we use NIQE [29] as the perceptual metric, while other metrics will be investigated in Section 4.4. Finally, we label every image pair to (3, 2, 1) according to the order of corresponding NIQE value (the one with the best NIQE value is set to 1).

Implementation details. As shown in Figure 2, we utilize VGG [33] structure to implement the Ranker [25], which includes 10 convolutional layers, a series of batch normalization and LeakyReLU operations. Instead of max-pooling, we apply convolutional layer with a kernel size 4 and stride 2 to downsample the features. In one iteration, two patches with different perceptual levels are randomly selected as the input of Ranker. For optimization, we use Adam [21] optimizer with weight decay 1 × 10⁻⁴. The learning rate is initialized to 1 × 10⁻³ and decreases with a factor 0.5 of every 10 × 10⁴ iterations for total 30 × 10⁴ iterations. The margin ϵ of margin-ranking loss is set to 0.5. For weight initialization, we use He. [17] method to initialize the weights of Ranker.

Evaluation. The Spearman Rank Order Correlation Coefficient (SROCC) [25] is a traditional evaluation metric to evaluate the performance of image quality assessment algorithms. In our experiment, SROCC is employed to measure the monotonic relationship between the label and the ranking score. Given N images, the SROCC is computed as:

\[ SROCC = 1 - \frac{6 \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{N(N^2 - 1)}, \]  

where \( y_i \) represents the order of label, and \( \hat{y}_i \) is the order of output score of the Ranker. SROCC has the ability to measure the accuracy of Ranker. The larger value of SROCC represents the better accuracy of Ranker. For validation dataset, the ranker achieves a SROCC of 0.88, which is an adequate performance compared with those in the related work [8, 25].

4.2. Training details of RankSRGAN

We use the DIV2K [1] dataset to train RankSRGAN. The patch sizes of HR and LR are set to 296 and 74, respectively. For testing, we use benchmark datasets Set14 [39], BSD100 [27] and PIRM-test [2]. PIRM-test is used to measure the perceptual quality of SR methods in PIRM2018-SR [2]. Following the settings of SRGAN [22], we employ a standard SRGAN [22] as our base model. The generator is built with 16 residual blocks, and the batch-normalization layers are removed [35]. The discriminator utilizes the VGG network [33] with ten convolutional layers. The mini-batch size is set to 8. At each training step, the combination of loss functions (Section 3.3) for the generator is:

\[ L_{\text{total}} = L_P + 0.005L_G + 0.03L_R, \]  

where the weights of \( L_G \) and \( L_R \) are determined empirically to obtain high perceptual improvement [8, 22, 35].
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metric</th>
<th>Bicubic</th>
<th>FSRCNN</th>
<th>SRResNet</th>
<th>SRGAN</th>
<th>ESRGAN</th>
<th>RankSRGAN (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set14</td>
<td>NIQE</td>
<td>7.61</td>
<td>6.92</td>
<td>6.12</td>
<td>3.82</td>
<td>3.28</td>
<td><strong>3.28</strong></td>
</tr>
<tr>
<td></td>
<td>PI</td>
<td>6.97</td>
<td>6.16</td>
<td>5.36</td>
<td>2.98</td>
<td>2.61</td>
<td><strong>2.61</strong></td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>26.08</td>
<td>27.66</td>
<td><strong>28.57</strong></td>
<td>26.68</td>
<td>26.39</td>
<td>26.57</td>
</tr>
<tr>
<td>BSD100</td>
<td>NIQE</td>
<td>7.60</td>
<td>7.11</td>
<td>6.43</td>
<td>3.29</td>
<td>3.21</td>
<td><strong>3.01</strong></td>
</tr>
<tr>
<td></td>
<td>PI</td>
<td>6.94</td>
<td>6.17</td>
<td>5.34</td>
<td>2.37</td>
<td>2.27</td>
<td><strong>2.15</strong></td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>25.96</td>
<td>26.94</td>
<td><strong>27.61</strong></td>
<td>25.67</td>
<td>25.72</td>
<td>25.57</td>
</tr>
<tr>
<td>PIRM-Test</td>
<td>NIQE</td>
<td>7.45</td>
<td>6.86</td>
<td>5.98</td>
<td>2.71</td>
<td>2.56</td>
<td><strong>2.51</strong></td>
</tr>
<tr>
<td></td>
<td>PI</td>
<td>7.33</td>
<td>6.02</td>
<td>5.18</td>
<td>2.09</td>
<td>1.98</td>
<td><strong>1.95</strong></td>
</tr>
<tr>
<td></td>
<td>PSNR</td>
<td>26.45</td>
<td>27.57</td>
<td><strong>28.33</strong></td>
<td>25.60</td>
<td>25.30</td>
<td>25.62</td>
</tr>
</tbody>
</table>

Table 2. Average NIQE [29], PI [2] and PSNR values on the Set14 [39], BSD100 [27] and PIRM-Test [2].

<table>
<thead>
<tr>
<th>Ground Truth</th>
<th>Bicubic</th>
<th>FSRCNN</th>
<th>SRResNet</th>
<th>SRGAN</th>
<th>ESRGAN</th>
<th>RankSRGAN (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Img 233</td>
<td>NIQE/PSNR</td>
<td>7.08/24.20</td>
<td>7.00/25.44</td>
<td>5.73/27.21</td>
<td>3.78/25.06</td>
<td>3.64/24.26</td>
</tr>
<tr>
<td>Img 157055</td>
<td>NIQE/PSNR</td>
<td>7.42/25.03</td>
<td>6.93/25.98</td>
<td>5.45/26.53</td>
<td>2.22/24.53</td>
<td>2.38/24.62</td>
</tr>
<tr>
<td>Img 106024</td>
<td>NIQE/PSNR</td>
<td>8.59/31.37</td>
<td>7.16/33.03</td>
<td>7.82/34.13</td>
<td>4.64/32.51</td>
<td>4.47/31.41</td>
</tr>
<tr>
<td>Img 242</td>
<td>NIQE/PSNR</td>
<td>7.76/26.37</td>
<td>6.75/27.02</td>
<td>5.86/27.75</td>
<td>3.98/26.18</td>
<td>3.58/25.88</td>
</tr>
</tbody>
</table>

Figure 5. Visual results comparison of our model with other works on 4 super-resolution. Lower NIQE value indicates better perceptual quality, while higher PSNR indicates less distortion.

Adam[21] optimization method with \( \beta_1 = 0.9 \) is used for training. For generator and discriminator, the initial learning rate is set to \( 1 \times 10^{-4} \) which is reduced by a half for multi-step \( [50 \times 10^3, 100 \times 10^3, 200 \times 10^3, 300 \times 10^3] \). A total of \( 600 \times 10^3 \) iterations are executed by PyTorch. In training procedure, we add Ranker to the standard SRGAN. The Ranker takes some time to predict the ranking score, thus the training time is a little slower (about 1.18 times) than standard SRGAN[22]. For the generator, the number of parameters remains the same as SRGAN[22].

4.3. Comparison with the-state-of-the-arts

We compare the performance of the proposed method with the state-of-the-art perceptual SR methods ESRGAN [35] and SRGAN [22] \(^1\). The evaluation metrics include NIQE [29], PI [2] and PSNR. Table 2 shows their performance on three test datasets – Set14, BSD100 and PIRM-Test. Note that lower NIQE/PI indicates better visual quality. When comparing our method with SRGAN and ESRGAN, we find that RankSRGAN achieves the best NIQE and PI performance on all test sets. Furthermore, the improvement of perceptual scores does not come at the price of PSNR. Note that in PIRM-Test, RankSRGAN also obtains the highest PSNR values among perceptual SR methods. Figure 5 shows some visual examples, where we observe that our method could generate more realistic textures without introducing additional artifacts (please see the win-

\(^1\) Our implementation of SRResNet and SRGAN achieve even better performance than that reported in the original paper.
As the results may vary across different iterations, we further show the convergence curves of RankSRGAN in Figure 6. Their performance on NIQE and PSNR are relatively stable during the training process. For PSNR, they obtain comparable results. But for NIQE, RankSRGAN is consistently better than SRGAN by a large margin.

### 4.4. Ablation study

**Effect of different rank datasets.** The key factor that influences the performance of Ranker is the choice of SR algorithms. In the main experiments, we use (SRResNet, SRGAN, ESRGAN) to generate the rank dataset. Then what if we select other SR algorithms? Will we always obtain better results than SRGAN? To answer the question, we first analyze the reason of using these three algorithms, then conduct another experiment using a different combination.

As our baseline model is SRGAN, we need the Ranker to have the ability to rank the outputs of SRGAN. Since the training of SRGAN starts from the pre-trained model SRResNet, the Ranker should recognize the results between SRResNet and SRGAN. That is the reason why we choose SRResNet and SRGAN. Then the next step is to find a better algorithm that could guide the model to achieve better results. We choose ESRGAN as it surpasses SRGAN by a large margin in the PIRM-SR2018 challenge [2]. Therefore, we believe that a better algorithm than SRGAN could always lead to better performance.

As we claim that Ranker can guide the SR model to be optimized in the direction of perceptual metrics, we need to verify whether it works for other perceptual metrics. We choose Ma [27] and PI [2], which show high correlation with Mean-Opinion-Score (Ma: 0.61, PI: 0.83) in [2]. We use Ma and PI as the evaluation metric to generate the rank dataset. All other settings remain the same as RankSRGAN with NIQE. The only difference in these experiments is the ranking labels in the rank dataset. The results are summarized in Table 4, where we observe that the Ranker could help RankSRGAN achieve the best performance in the chosen metric. This shows that our method can generalize well on different perceptual metrics.

As our baseline model is SRGAN, we need the Ranker to have the ability to rank the outputs of SRGAN. Since the training of SRGAN starts from the pre-trained model SRResNet, the Ranker should recognize the results between SRResNet and SRGAN. That is the reason why we choose SRResNet and SRGAN. Then the next step is to find a better algorithm that could guide the model to achieve better results. We choose ESRGAN as it surpasses SRGAN by a large margin in the PIRM-SR2018 challenge [2]. Therefore, we believe that a better algorithm than SRGAN could always lead to better performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>SRGAN</th>
<th>RankSRGAN</th>
<th>RankSRGAN-HR</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIQE</td>
<td>2.70</td>
<td>2.51</td>
<td>2.58</td>
</tr>
<tr>
<td>PSNR</td>
<td>25.62</td>
<td>25.60</td>
<td><strong>26.00</strong></td>
</tr>
</tbody>
</table>

Table 3. Comparison with RankSRGAN and RankSRGAN-HR.

To validate this comment, we directly use the ground truth HR as the third algorithm, which is the extreme case. We still apply NIQE for evaluation. Interestingly, although HR images have infinite PSNR values, they cannot surpass all the results of SRGAN on NIQE. Similar to ESRGAN, HR and SRGAN have mixed ranking orders. We train our Ranker with (SRResNet, SRGAN, HR) and obtain the new SR model – RankSRGAN-HR. Table 3 compares its results with SRGAN and RankSRGAN. As expected, RankSRGAN-HR achieves better NIQE values than SRGAN. But at the same time, RankSRGAN-HR also improves the PSNR by almost 0.4 dB. It achieves a good balance between the perceptual metric and PSNR. This also indicates that the model could always have an improvement space as long as we have better algorithms for guidance.

**Effect of different perceptual metrics.** As we claim that Ranker can guide the SR model to be optimized in the direction of perceptual metrics, we need to verify whether it works for other perceptual metrics. We choose Ma [27] and PI [2], which show high correlation with Mean-Opinion-Score (Ma: 0.61, PI: 0.83) in [2]. We use Ma and PI as the evaluation metric to generate the rank dataset. All other settings remain the same as RankSRGAN with NIQE. The only difference in these experiments is the ranking labels in the rank dataset. The results are summarized in Table 4, where we observe that the Ranker could help RankSRGAN achieve the best performance in the chosen metric. This shows that our method can generalize well on different perceptual metrics.

Effect of different rank datasets. The key factor that influences the performance of Ranker is the choice of SR algorithms. In the main experiments, we use (SRResNet, SRGAN, ESRGAN) to generate the rank dataset. Then what if we select other SR algorithms? Will we always obtain better results than SRGAN? To answer the question, we first analyze the reason of using these three algorithms, then conduct another experiment using a different combination.

As our baseline model is SRGAN, we need the Ranker to have the ability to rank the outputs of SRGAN. Since the training of SRGAN starts from the pre-trained model SRResNet, the Ranker should recognize the results between SRResNet and SRGAN. That is the reason why we choose SRResNet and SRGAN. Then the next step is to find a better algorithm that could guide the model to achieve better results. We choose ESRGAN as it surpasses SRGAN by a large margin in the PIRM-SR2018 challenge [2]. Therefore, we believe that a better algorithm than SRGAN could always lead to better performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>NIQE</th>
<th>10-Ma</th>
<th>PI</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGAN</td>
<td>2.71</td>
<td>1.47</td>
<td>2.09</td>
<td>25.62</td>
</tr>
<tr>
<td>ESRGAN</td>
<td>2.56</td>
<td>1.40</td>
<td>1.98</td>
<td>25.30</td>
</tr>
<tr>
<td>RankSRGAN_N</td>
<td><strong>2.51</strong></td>
<td>1.39</td>
<td>1.95</td>
<td>25.62</td>
</tr>
<tr>
<td>RankSRGAN_M</td>
<td>2.65</td>
<td><strong>1.38</strong></td>
<td>2.01</td>
<td>25.21</td>
</tr>
<tr>
<td>RankSRGAN_P</td>
<td>2.49</td>
<td>1.39</td>
<td><strong>1.94</strong></td>
<td>25.49</td>
</tr>
</tbody>
</table>

Table 4. The performance of RankSRGAN with different Rankers. N: Ranker with NIQE [29], M: Ranker with Ma [26] and P: Ranker with PI [2].

**Effect of Ranker: Rank VS. Regression.** To train our Ranker, we choose to use the ranking orders instead of the real values of the perceptual metric. Actually, we can also let the network directly learn the real values. In [8], Choi et al. use a regression network to predict a subjective score for a given image and define a corresponding subjective score loss. To compare these two strategies, we train a “regression” Ranker with MSE loss instead of the margin-ranking loss. The labels in the rank dataset are real values of the perceptual metric. We use NIQE and Ma to generate the labels of rank dataset. All the other settings remain the same as RankSRGAN.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Method</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIQE</td>
<td>regression</td>
<td>0.06</td>
</tr>
<tr>
<td>NIQE</td>
<td>rank</td>
<td><strong>0.11</strong></td>
</tr>
<tr>
<td>Ma</td>
<td>regression</td>
<td>0.09</td>
</tr>
<tr>
<td>Ma</td>
<td>rank</td>
<td><strong>0.15</strong></td>
</tr>
</tbody>
</table>

Table 5. The distance between SR1 and SR2 with regression and rank.

Theoretically, the real values of perceptual metrics may distribute unevenly among different algorithms. For example, SRGAN and ESRGAN are very close to each other on NIQE values. This presents a difficulty for the learning of regression. On the contrary, learning ranking orders
can simply ignore these variances. In experiments, we first measure the distances between the outputs of SRGAN and ESRGAN with different strategies. Table 5 shows the mean absolute distances of these two strategies. Obviously, results with rank have larger distances than results with regression. When applying these Rankers in SR training, the rank strategy achieves better performance than the regression strategy on the selected perceptual metric. Results are shown in Table 6.

<table>
<thead>
<tr>
<th>Method</th>
<th>NIQE</th>
<th>10-Ma</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGAN</td>
<td>2.71</td>
<td>1.47</td>
<td>25.62</td>
</tr>
<tr>
<td>ESRGAN</td>
<td>2.55</td>
<td>1.40</td>
<td>25.30</td>
</tr>
<tr>
<td>RankSRGAN-Re_N</td>
<td>2.53</td>
<td>1.42</td>
<td>25.58</td>
</tr>
<tr>
<td>RankSRGAN_N</td>
<td>2.51</td>
<td>1.39</td>
<td>25.60</td>
</tr>
<tr>
<td>RankSRGAN-Re_M</td>
<td>2.61</td>
<td>1.43</td>
<td>25.23</td>
</tr>
<tr>
<td>RankSRGAN_M</td>
<td>2.65</td>
<td>1.38</td>
<td>25.21</td>
</tr>
</tbody>
</table>

Table 6. The performance of RankSRGAN with different rankers. Re: Ranker with regression, N: Ranker with NIQE, M: Ranker with Ma.

Effect of different losses. To test the effects of rank-content loss, we expect to add MSE loss to achieve improvement in PSNR. Table 7 shows the performance of our method trained with the combination of loss functions.

<table>
<thead>
<tr>
<th>Method</th>
<th>Loss</th>
<th>NIQE</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGAN</td>
<td>$L_P$</td>
<td>2.71</td>
<td>25.62</td>
</tr>
<tr>
<td>ESRGAN</td>
<td>$L_P + 10L_M$</td>
<td>2.55</td>
<td>25.30</td>
</tr>
<tr>
<td>RankSRGAN</td>
<td>$L_P + L_R$</td>
<td>2.51</td>
<td>25.62</td>
</tr>
<tr>
<td>RankSRGAN-M_1</td>
<td>$L_P + L_R + \alpha_1 L_M$</td>
<td>2.55</td>
<td>25.87</td>
</tr>
<tr>
<td>RankSRGAN-M_2</td>
<td>$L_P + L_R + \alpha_2 L_M$</td>
<td>2.72</td>
<td>26.62</td>
</tr>
</tbody>
</table>

Table 7. The performance of RankSRGAN with different rankers. L: perceptual loss, R: rank-content loss, M: MSE loss. $\alpha_1, \alpha_2 \in \{1, 5\}$

As expected, increasing the contribution of the MSE loss with the larger $\alpha$ results in higher PSNR values. On the other hand, the NIQE values are increased which is a trade-off between PSNR and NIQE as mentioned in [3], and our method has a capability to deal with the priorities by adjusting the weights of the loss functions.

4.5. User study

To demonstrate the effectiveness and superiority of RankSRGAN, we conduct a user study against state-of-the-art models, i.e. SRGAN [22] and ESRGAN [35]. In the first session, two different SR images are shown at the same time where one is generated by the proposed RankSRGAN and the other is generated by SRGAN or ESRGAN. The participants are required to pick the image that is more visually pleasant (more natural and realistic). We use the PIRM-Test [2] dataset as the testing dataset. There are a total of 100 images, from which 30 images are randomly selected for each participant. To make a better comparison, one small patch from the image is zoomed in. In the second session, we focus on the perceptual quality of different typical SR methods in a sorting manner. The participants are asked to rank 4 versions of each image: SRResNet [22], ESRGAN [35], RankSRGAN, and the Ground Truth (GT) image according to their visual qualities. Similar to the first session, 20 images are randomly shown for each participant. There are totally 30 participants to finish the user study.

As suggested in Figure 7, RankSRGAN has achieved better visual performance against ESRGAN and SRGAN. Since RankSRGAN consists of a base model SRGAN and the proposed Ranker, it can naturally inherit the characteristics of SRGAN and achieve better performance in perceptual metric. Thus, RankSRGAN performs more similar to SRGAN than ESRGAN. Figure 8 shows the ranking results of different SR methods. As RankSRGAN has the best performance in perceptual metric, the ranking results of RankSRGAN are second to GT images, but sometimes it even produces images comparable to GT.

5. Conclusion

For perceptual super-resolution, we propose RankSRGAN to optimize SR model in the orientation of perceptual metrics. The key idea is introducing a Ranker to learn the behavior of the perceptual metrics by learning to rank approach. Moreover, our proposed method can combine the strengths of different SR methods and generate better results. Extensive experiments well demonstrate that our RankSRGAN is a flexible framework, which can achieve superiority over state-of-the-art methods in perceptual metric and have the ability to recover more realistic textures.

Acknowledgements. This work is partially supported by National Natural Science Foundation of China (61876176, U1613211), Shenzhen Basic Research Program (JCYJ20170818164704758), the Joint Lab of CAS-HK.
References


