Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data

Xintao Wang[1] Liangbin Xie*[2,3] Chao Dong[2,4] Ying Shan[1]
[1]Applied Research Center (ARC), Tencent PCG
[2]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[3]University of Chinese Academy of Sciences [4]Shanghai AI Laboratory
{xintaowang, yingsshan}@tencent.com {lb.xie, chao.dong}@siat.ac.cn
https://github.com/xinntao/Real-ESRGAN

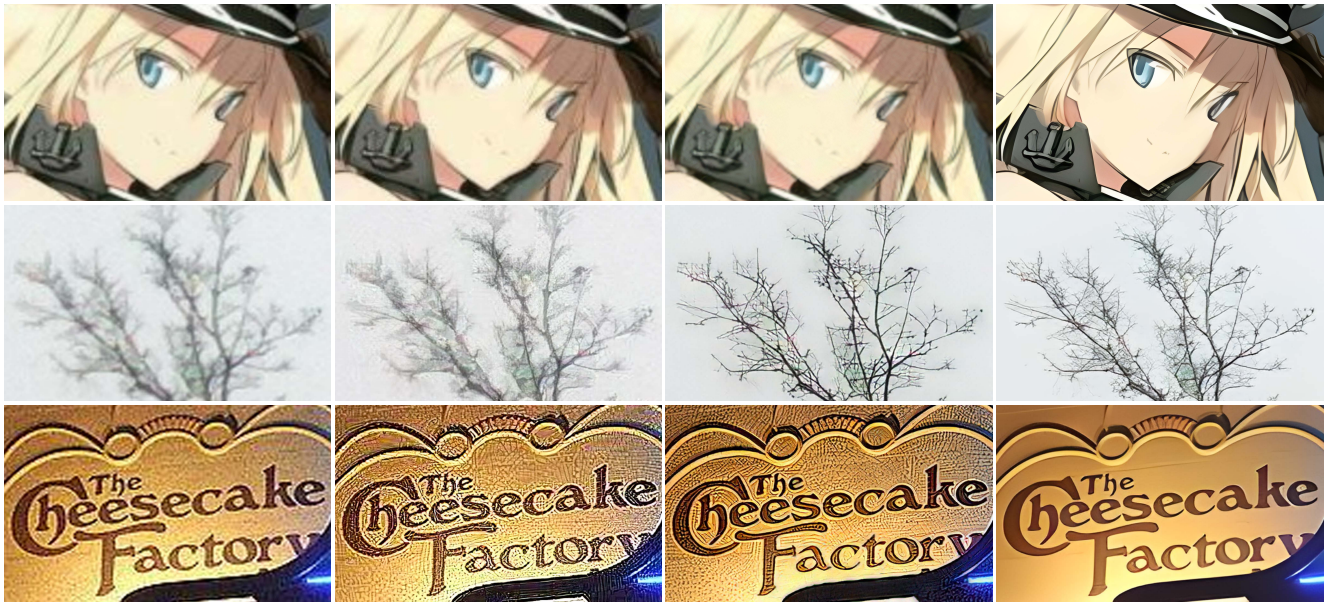

Figure 1: Comparisons of bicubic-upsampled, ESRGAN [50], RealSR [19], and our Real-ESRGAN results on real-life images. The Real-ESRGAN model trained with pure synthetic data is capable of enhancing details while removing annoying artifacts for common real-world images. (**Zoom in for best view**)

Abstract

Though many attempts have been made in blind super-resolution to restore low-resolution images with unknown and complex degradations, they are still far from addressing general real-world degraded images. In this work, we extend the powerful ESRGAN to a practical restoration application (namely, Real-ESRGAN), which is trained with pure synthetic data. Specifically, a high-order degradation modeling process is introduced to better simulate complex real-world degradations. We also consider the common ringing and overshoot artifacts in the synthesis process. In addition, we employ a U-Net discriminator with spectral normalization to increase discriminator capability and stabilize the training dynamics. Extensive comparisons have shown its superior visual performance than prior works on various real datasets. We also provide efficient implementations to synthesize training pairs on the fly.

1. Introduction

Single image super-resolution (SR) [13, 10, 27] is an active research topic, which aims at reconstructing a high-resolution (HR) image from its low-resolution (LR) counterpart. Since the pioneering work of SRCNN [9], deep convolution neural network (CNN) approaches have brought prosperous developments in the SR field. However, most approaches [21, 27, 20, 25, 50] assume an *ideal bicubic downsampling kernel*, which is different from real degradations. This degradation mismatch makes those approaches unpractical in real-world scenarios.

Blind super-resolution [35, 2, 56], on the contrary, aims to restore low-resolution images suffering from *unknown and complex degradations*. Existing approaches can be roughly categorized into *explicit modeling* and *implicit modeling*, according to the underlying degradation process. Classical degradation model [11, 29], which consists of blur, downsampling, noise and JPEG compression (more details in Sec. 3.1), is widely adopted in explicit model-

*Liangbin Xie is an intern in Applied Research Center, Tencent PCG

ing methods [56, 16, 34]. However, the real-world degradations are usually too complex to be modeled with a simple combination of multiple degradations. Thus, these methods will easily fail in real-world samples. Implicit modeling methods [54, 12, 46] utilize data distribution learning with Generative Adversarial Network (GAN) [14] to obtain the degradation model. Yet, they are limited to the degradations within training datasets, and could not generalize well to out-of-distribution images. Readers are encouraged to refer to a recent blind SR survey [28] for a more comprehensive taxonomy.

In this work, we aim to extend the powerful ESRGAN [50] to restore general real-world LR images by synthesizing training pairs with a more practical degradation process. The real complex degradations usually come from *complicate combinations of different degradation processes*, such as imaging system of cameras, image editing, and Internet transmission. For example, when we take a photo with our cellphones, the photos may have several degradations, such as camera blur, sensor noise, sharpening artifacts, and JPEG compression. We then do some editing and upload to a social media app, which introduces further compression and unpredictable noises. The above process becomes more complicated when the image is shared several times on the Internet.

This motivates us to extend the *classical "first-order" degradation model* to **"high-order" degradation modeling** for real-world degradations, *i.e.*, the degradations are modeled with several repeated degradation processes, each process being the classical degradation model. Empirically, we adopt a *second-order degradation process* for a good balance between simplicity and effectiveness. A recent work [55] also proposes a random shuffling strategy to synthesize more practical degradations. However, it still involves a fixed number of degradation processes, and whether all the shuffled degradations are useful or not is unclear. Instead, high-order degradation modeling is more flexible and attempts to mimic the real degradation generation process. We further incorporate $sinc$ filters in the synthesis process to simulate the **common ringing and overshoot artifacts**.

As the degradation space is much larger than ESRGAN, the training also becomes challenging. Specifically, 1) the discriminator requires a more powerful capability to discriminate realness from complex training outputs, while the gradient feedback from the discriminator needs to be more accurate for local detail enhancement. Therefore, we improve the VGG-style discriminator in ESRGAN to an **U-Net design** [41, 52, 39]. 2) The U-Net structure and complicate degradations also increase the training instability. Thus, we employ the **spectral normalization (SN) regularization** [37, 41] to stabilize the training dynamics. Equipped with the dedicated improvements, we are able to easily train our Real-ESRGAN and achieve a good balance of local detail enhancement and artifact suppression.

To summarize, in this work, **1)** we propose a high-order degradation process to model practical degradations, and utilize $sinc$ filters to model common ringing and overshoot artifacts. **2)** We employ several essential modifications (*e.g.*, U-Net discriminator with spectral normalization) to increase discriminator capability and stabilize the training dynamics. **3)** Real-ESRGAN trained with pure synthetic data is able to restore most real-world images and achieve better visual performance than previous works, making it more practical in real-world applications.

2. Related Work

The image super-resolution field [21, 24, 45, 17, 25, 27, 58, 22, 44, 57, 7, 30] has witnessed a variety of developments since SRCNN [9, 10]. To achieve visually-pleasing results, generative adversarial network [15] is usually employed as loss supervisions to push the solutions closer to the natural manifold [26, 40, 50, 49]. Most methods assume a bicubic downsampling kernel and usually fail in real images. Recent works also incorporate reinforcement learning or GAN prior to image restoration [53, 6, 47].

There have been several excellent explorations in blind SR. The first category involves explicit degradation representations and typically consists of two components: degradation prediction and conditional restoration. The above two components are performed either separately [2, 56] or jointly (iteratively) [16, 34, 46]. These approaches rely on predefined degradation representations (*e.g.*, degradation types and levels), and usually consider simple synthetic degradations. Moreover, inaccurate degradation estimations will inevitably result in artifacts.

Another category is to obtain/generate training pairs as close to real data as possible, and then train a unified network to address blind SR. The training pairs are usually 1) captured with specific cameras followed by tedious alignments [5, 51]; 2) or directly learned from unpaired data with cycle consistency loss [54, 33]; 3) or synthesized with estimated blur kernels and extracted noise patches [60, 19]. However, 1) the captured data is only constrained to degradations associated with specific cameras, and thus could not well generalize to other real images; 2) learning fine-grained degradations with unpaired data is challenging, and the results are usually unsatisfactory.

Degradation models. Classical degradation model [11, 29] is widely adopted in blind SR methods [56, 16, 34]. Yet, real-world degradations are usually too complex to be explicitly modeled. Thus, implicit modeling attempts to learn a degradation generation process within networks [54, 12, 46]. In this work, we propose a flexible high-order degradation model to synthesize more practical degradations.

3. Methodology

3.1. Classical Degradation Model

Blind SR aims to restore high-resolution images from low-resolution ones with unknown and complex degradations. The classical degradation model [11, 29] is usually adopted to synthesize the low-resolution input. Generally, the ground-truth image $\boldsymbol{y}$ is first convolved with blur kernel $\boldsymbol{k}$. Then, a downsampling operation with scale factor r is performed. The low-resolution $\boldsymbol{x}$ is obtained by adding noise $\boldsymbol{n}$. Finally, JPEG compression is also adopted, as it is widely-used in real-world images.

$$\boldsymbol{x} = \mathcal{D}(\boldsymbol{y}) = [(\boldsymbol{y} \circledast \boldsymbol{k}) \downarrow_r + \boldsymbol{n}]_{\texttt{JPEG}}, \tag{1}$$

where $\mathcal{D}$ denotes the degradation process. In the following, we briefly revisit these commonly-used degradations. The detailed settings are specified in Sec. 4.1. More descriptions and examples are in Appendix. A.

Blur. We typically model blur degradation as a convolution with a linear blur filter (kernel). Isotropic and anisotropic Gaussian filters are common choices. For a Gaussian blur kernel $\boldsymbol{k}$ with a kernel size of $2t+1$, its $(i,j) \in [-t,t]$ element is sampled from a Gaussian distribution, formally:

$$\boldsymbol{k}(i,j) = \frac{1}{N}\exp(-\frac{1}{2}\boldsymbol{C}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{C}), \quad \boldsymbol{C} = [i,j]^T, \tag{2}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix; $\boldsymbol{C}$ is the spatial coordinates; N is the normalization constant. The covariance matrix could be further represented as follows:

$$\boldsymbol{\Sigma} = \boldsymbol{R}\begin{bmatrix}\sigma_1^2 & 0\\ 0 & \sigma_2^2\end{bmatrix}\boldsymbol{R}^T, \quad (\boldsymbol{R} \text{ is the rotation matrix}) \tag{3}$$

$$= \begin{bmatrix}cos\theta & -sin\theta\\ sin\theta & cos\theta\end{bmatrix}\begin{bmatrix}\sigma_1^2 & 0\\ 0 & \sigma_2^2\end{bmatrix}\begin{bmatrix}cos\theta & sin\theta\\ -sin\theta & cos\theta\end{bmatrix}, \tag{4}$$

where σ_1 and σ_2 are the standard deviation along the two principal axes (*i.e.*, eigenvalues of the covariance matrix); θ is the rotation degree. When $\sigma_1 = \sigma_2$, $\boldsymbol{k}$ is an isotropic Gaussian blur kernel; otherwise $\boldsymbol{k}$ is an anisotropic kernel.

Discussion. Though Gaussian blur kernels are widely used to model blur degradation, they may not well approximate real camera blur. To include more diverse kernel shapes, we further adopt generalized Gaussian blur kernels [31] and a plateau-shaped distribution. Their probability density function (pdf) are $\frac{1}{N}\exp(-\frac{1}{2}(\boldsymbol{C}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{C})^{\beta}$, and $\frac{1}{N}\frac{1}{1+(\boldsymbol{C}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{C})^{\beta}}$, respectively. β is the shape parameter. Empirically, we find that including these blur kernels could produce sharper outputs for several real samples.

Noise. We consider two commonly-used noise types – 1) additive Gaussian noise and 2) Poisson noise. Addictive Gaussian noise has a probability density function equal to that of the Gaussian distribution. The noise intensity is controlled by the standard deviation (*i.e.*, sigma value) of the Gaussian distribution. When each channel of RGB images has independent sampled noise, the synthetic noise is color noise. We also synthesize gray noise by employing the same sampled noise to all three channels [55, 38].

Poisson noise follows the Poisson distribution. It is usually used to approximately model the sensor noise caused by statistical quantum fluctuations, that is, variation in the number of photons sensed at a given exposure level. Poisson noise has an intensity proportional to the image intensity, and the noises at different pixels are independent.

Resize (Downsampling). Downsampling is a basic operation for synthesizing low-resolution images in SR. More generally, we consider both downsamping and upsampling, *i.e.*, the resize operation. There are several resize algorithms - nearest-neighbor interpolation, area resize, bilinear interpolation, and bicubic interpolation. Different resize operations bring in different effects - some produce blurry results while some may output over-sharp images with overshoot artifacts.

In order to include more diverse and complex resize effects, we consider a random resize operation from the above choices. As nearest-neighbor interpolation introduces the misalignment issue, we exclude it and only consider the area, bilinear and bicubic operations.

JPEG compression. JPEG compression is a commonly used technique of lossy compression for digital images. It first converts images into the YCbCr color space and downsamples the chroma channels. Images are then split into 8×8 blocks and each block is transformed with a two-dimensional discrete cosine transform (DCT), followed by a quantization of DCT coefficients. More details of JPEG compression algorithms can be found in [43]. Unpleasing block artifacts are usually introduced by the JPEG compression.

The quality of compressed images is determined by a quality factor $q \in [0, 100]$, where a lower q indicates a higher compression ratio and worse quality. We use the PyTorch implementation - `DiffJPEG` [32].

3.2. High-order Degradation Model

When we adopt the above classical degradation model to synthesize training pairs, the trained model could indeed handle some real samples. However, it still can not resolve some complicated degradations in the real world, especially the unknown noises and complex artifacts (see Fig. 3). It is because that the synthetic low-resolution images still have a large gap with realistic degraded images. We thus extend the classical degradation model to a high-order degradation process to model more practical degradations.

The classical degradation model only includes a fixed

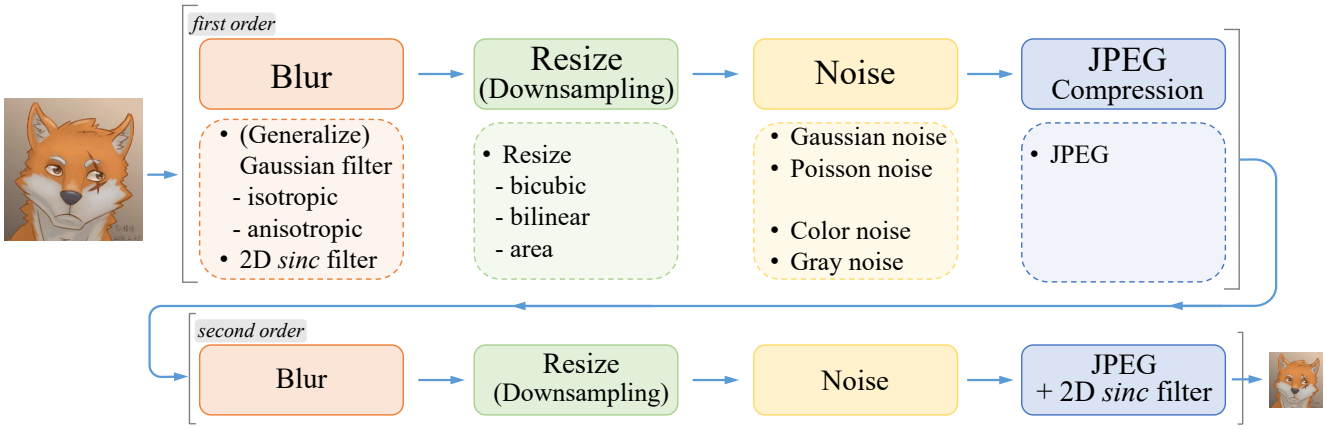

Figure 2: Overview of the pure synthetic data generation adopted in Real-ESRGAN. It utilizes a second-order degradation process to model more practical degradations, where each degradation process adopts the classical degradation model. The detailed choices for blur , resize, noise and JPEG compression are listed. We also employ $sinc$ filter to synthesize common ringing and overshoot artifacts.

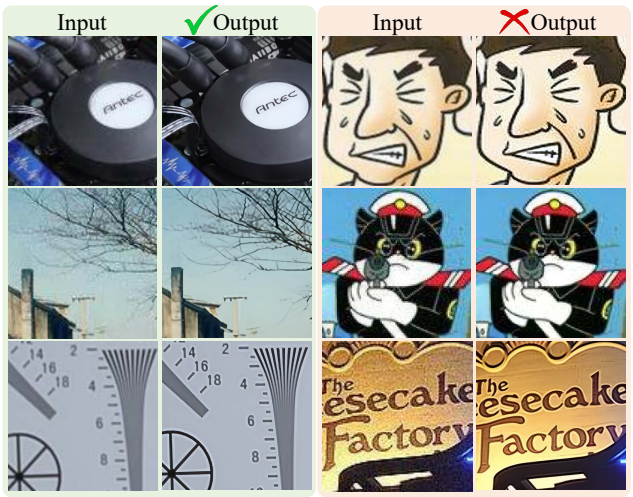

Figure 3: Models trained with synthetic data of classical degradation model could resolve some real samples (Left). Yet, they amplify noises or introduce ringing artifacts for complex real-world images (Right). Zoom in for best view

number of basic degradations, which can be regarded as a first-order modeling. However, the real-life degradation processes are quite diverse, and usually comprise a series of procedures including imaging system of cameras, image editing, Internet transmission, *etc*. For instance, when we want to restore a low-quality image download from the Internet, its underlying degradation involves a complicated combination of different degradation processes. Specifically, the original image might be taken with a cellphone many years ago, which inevitably contains degradations such as camera blur, sensor noise, low resolution and JPEG compression. The image was then edited with sharpening and resize operations, bringing in overshoot and blur artifacts. After that, it was uploaded to some social media applications, which introduces a further compression and unpredictable noises. As the digital transmission will also bring artifacts, this process becomes more complicated when the image spreads several times on the Internet.

Such a complicated deterioration process could not be modeled with the classical first-order model. Thus, we propose a high-order degradation model. An n-order model involves n repeated degradation processes (as shown in Eq. 5), where each degradation process adopts the classical degradation model (Eq. 1) with the same procedure but different hyper-parameters. Note that the "high-order" here is different from that used in mathematical functions. It mainly refers to the implementation time of the same operation. The random shuffling strategy in [55] may also include repeated degradation processes (*e.g.*, double blur or JPEG). But we highlight that the high-order degradation process is the key, indicating that not all the shuffled degradations are necessary. In order to keep the image resolution in a reasonable range, the downsampling operation in Eq. 1 is replaced with a random resize operation. Empirically, we adopt a second-order degradation process, as it could resolve most real cases while keeping simplicity. Fig. 2 depicts the overall pipeline of our pure synthetic data generation pipeline.

$$\boldsymbol{x} = \mathcal{D}^n(\boldsymbol{y}) = (\mathcal{D}_n \circ \cdots \circ \mathcal{D}_2 \circ \mathcal{D}_1)(\boldsymbol{y}). \tag{5}$$

It is worth noting that the improved high-order degradation process is not perfect and could not cover the whole degradation space in the real world. Instead, it merely extends the solvable degradation boundary of previous blind SR methods through modifying the data synthesis process. Several typical limitation scenarios can be found in Fig. 11.

3.3. Ringing and overshoot artifacts

Ringing artifacts often appear as spurious edges near sharp transitions in an image. They visually look like bands or "ghosts" near edges. Overshoot artifacts are usually combined with ringing artifacts, which manifest themselves as an increased jump at the edge transition. The main cause of these artifacts is that the signal is bandlimited without high frequencies. These artifacts are very common and usually

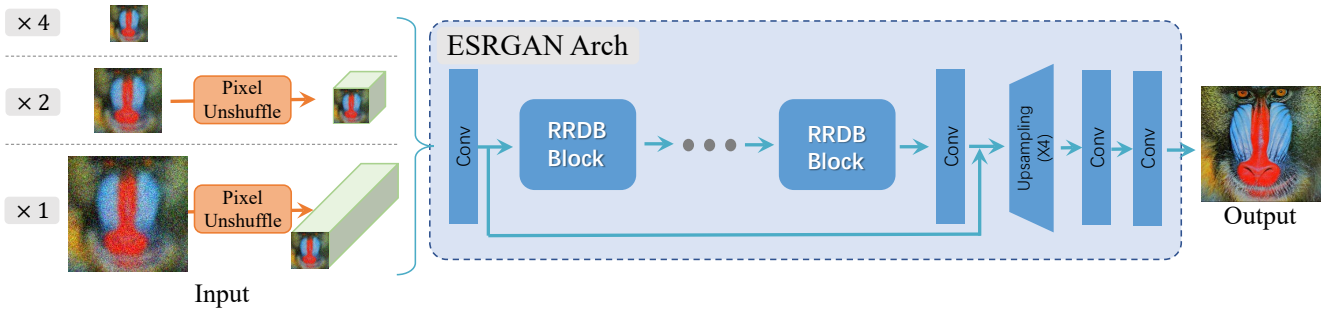

Figure 4: Real-ESRGAN adopts the same generator network as that in ESRGAN. For the scale factor of $\times 2$ and $\times 1$, it first employs a pixel-unshuffle operation to reduce spatial size and re-arrange information to the channel dimension.

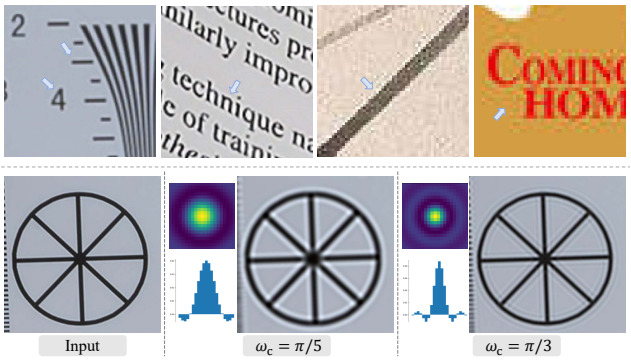

Figure 5: **Top**: Real samples suffering from ringing and overshoot artifacts. **Bottom**: Examples of *sinc* kernels (kernel size 21) and the corresponding filtered images. Zoom in for best view

produced by a sharping algorithm, JPEG compression, *etc*. Fig. 5 (Top) shows some real samples suffering from ringing and overshoot artifacts.

We employ the *sinc* filter, an idealized filter that cuts off high frequencies, to synthesize ringing and overshoot artifacts for training pairs. The *sinc* filter kernel can be expressed as[1]:

$$\boldsymbol{k}(i,j) = \frac{\omega_c}{2\pi\sqrt{i^2+j^2}} J_1(\omega_c\sqrt{i^2+j^2}), \qquad (6)$$

where (i,j) is the kernel coordinate; ω_c is the cutoff frequency; and J_1 is the first order Bessel function of the first kind. Fig. 5 (Bottom) shows *sinc* filters with different cutoff frequencies, and their corresponding filtered images. It is observed that it could well synthesize ringing and overshoot artifacts (especially introduced by over-sharp effects). These artifacts are visually similar to those in the first two real samples in Fig. 5 (Top).

We adopt *sinc* filters in two places: the blurring process and the last step of the synthesis. The order of the last *sinc* filter and JPEG compression is randomly exchanged to cover a larger degradation space, as some images may be first over-sharpened (with overshoot artifacts) and then have

[1]We use the implementation in this url.

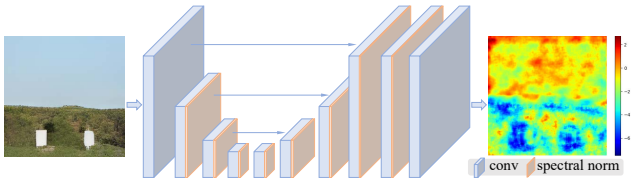

Figure 6: Architecture of the U-Net discriminator with spectral normalization.

JPEG compression; while some images may first do JPEG compression followed by sharpening operation.

3.4. Networks and Training

ESRGAN generator. We adopt the same generator (SR network) as ESRGAN [50], *i.e*., a deep network with several residual-in-residual dense blocks (RRDB), as shown in Fig. 4. We also extend the original $\times 4$ ESRGAN architecture to perform super-resolution with a scale factor of $\times 2$ and $\times 1$. As ESRGAN is a heavy network, we first employ the pixel-unshuffle (an inverse operation of pixel-shuffle [42]) to reduce the spatial size and enlarge the channel size before feeding inputs into the main ESRGAN architecture. Thus, the most calculation is performed in a smaller resolution space, which can reduce the GPU memory and computational resources consumption.

U-Net discriminator with spectral normalization (SN). As Real-ESRGAN aims to address a much larger degradation space than ESRGAN, the original design of discriminator in ESRGAN is no longer suitable. Specifically, the discriminator in Real-ESRGAN requires a greater discriminative power for complex training outputs. Instead of discriminating global styles, it also needs to produce accurate gradient feedback for local textures. Inspired by [41, 52], we also improve the VGG-style discriminator in ESRGAN to an U-Net design with skip connections (Fig. 6). The U-Net outputs realness values for each pixel, and can provide detailed per-pixel feedback to the generator.

In the meanwhile, the U-Net structure and complicate degradations also increase the training instability. We employ the spectral normalization regularization [37] to stabilize the training dynamics. Moreover, we observe that spectral normalization is also beneficial to alleviate the over-

sharp and annoying artifacts introduced by GAN training. With those adjustments, we are able to easily train the Real-ESRGAN and achieve a good balance of local detail enhancement and artifact suppression.

The training process is divided into two stages. First, we train a PSNR-oriented model with the L1 loss. The obtained model is named by *Real-ESRNet*. We then use the trained PSNR-oriented model as an initialization of the generator, and train the *Real-ESRGAN* with a combination of L1 loss, perceptual loss [20] and GAN loss [14, 26, 4].

4. Experiments

4.1. Datasets and Implementation

Training details. Similar to ESRGAN, we adopt DIV2K [1], Flickr2K [45] and OutdoorSceneTraining [49] datasets for training. The training HR patch size is set to 256. We train our models with four NVIDIA V100 GPUs with a total batch size of 48. We employ Adam optimizer [23]. Real-ESRNet is finetuned from ESRGAN for faster convergence. We train Real-ESRNet for $1000K$ iterations with learning rate 2×10^{-4} while training Real-ESRGAN for $400K$ iterations with learning rate 1×10^{-4}. We adopt exponential moving average (EMA) for more stable training and better performance. Real-ESRGAN is trained with a combination of L1 loss, perceptual loss and GAN loss, with weights $\{1, 1, 0.1\}$, respectively. We use the $\{\texttt{conv1}, ...\texttt{conv5}\}$ feature maps (with weights $\{0.1, 0.1, 1, 1, 1\}$) before activation in the pre-trained VGG19 network [20] as the perceptual loss. Our implementation is based on the BasicSR [48].

Degradation details. We employ a second-order degradation model for a good balance of simplicity and effectiveness. Unless otherwise specified, the two degradation processes have the same settings. We adopt Gaussian kernels, generalized Gaussian kernels and plateau-shaped kernels, with a probability of $\{0.7, 0.15, 0.15\}$. The blur kernel size is randomly selected from $\{7, 9, ...21\}$. Blur standard deviation σ is sampled from $[0.2, 3]$ ($[0.2, 1.5]$ for the second degradation process). Shape parameter β is sampled from $[0.5, 4]$ and $[1, 2]$ for generalized Gaussian and plateau-shaped kernels, respectively. We also use $sinc$ kernel with a probability of 0.1. We skip the second blur degradation with a probability of 0.2.

We employ Gaussian noises and Poisson noises with a probability of $\{0.5, 0.5\}$. The noise sigma range and Poisson noise scale are set to $[1, 30]$ and $[0.05, 3]$, respectively ($[1, 25]$ and $[0.05, 2.5]$ for the second degradation process). The gray noise probability is set to 0.4. JPEG compression quality factor is set to $[30, 95]$. The final $sinc$ filter is applied with a probability of 0.8. More details can be found in the released codes.

Training pair pool. In order to improve the training efficiency, all degradation processes are implemented in PyTorch with CUDA acceleration, so that we are able to synthesize training pairs on the fly. However, batch processing limits the diversity of synthetic degradations in a batch. For example, samples in a batch could not have different resize scaling factors. Therefore, we employ a training pair pool to increase the degradation diversity in a batch. At each iteration, the training samples are randomly selected from the training pair poor to form a training batch. We set the pool size to 180 in our implementation.

Sharpen ground-truth images during training. We further show a training trick to visually improve the sharpness, while not introducing visible artifacts. A typical way of sharpening images is to employ a post-process algorithm, such as unsharp masking (USM). However, this algorithm tends to introduce overshoot artifacts. We empirically find that sharpening ground-truth images during training could achieve a better balance of sharpness and overshoot artifact suppression. We denote the model trained with sharped ground-truth images as Real-ESRGAN**+** (comparisons are shown in Fig. 7).

4.2. Comparisons with Prior Works

We compare our Real-ESRGAN with several state-of-the-art methods, including ESRGAN [50], DAN [34], CDC [51], RealSR [19] and BSRGAN [55]. We test on several diverse testing datasets with real-world images, including RealSR [5], DRealSR [51], OST300 [49], DPED [18], ADE20K validation [59] and images from Internet. Since existing metrics for perceptual quality cannot well reflect the actual human perceptual preferences on the fine-grained scale [3], we present several representative visual samples in Fig. 7. The quantitative results are also included in the Appendix. B for reference.

It can be observed from Fig. 7 that our Real-ESRGAN outperforms previous approaches in both removing artifacts and restoring texture details. Real-ESRGAN+ (trained with sharpened ground-truths) can further boost visual sharpness. Specifically, the first sample contains overshoot artifacts (white edges around letters). Directly upsampling will inevitably amplify those artifacts (*e.g.*, DAN and BSRGAN). Real-ESRGAN takes such common artifacts into consideration and simulates them with $sinc$ filter, thus effectively removing ringing and overshoot artifacts. The second sample contains unknown and complicated degradations. Most algorithms can not effectively eliminate them while Real-ESRGAN trained with second-order degradation processes could. Real-ESRGAN is also capable of restoring more realistic textures (*e.g.*, brick, mountain and tree textures) for real-world samples, while other methods either fail to remove degradations or add unnatural textures (*e.g.*, RealSR and BSRGAN).

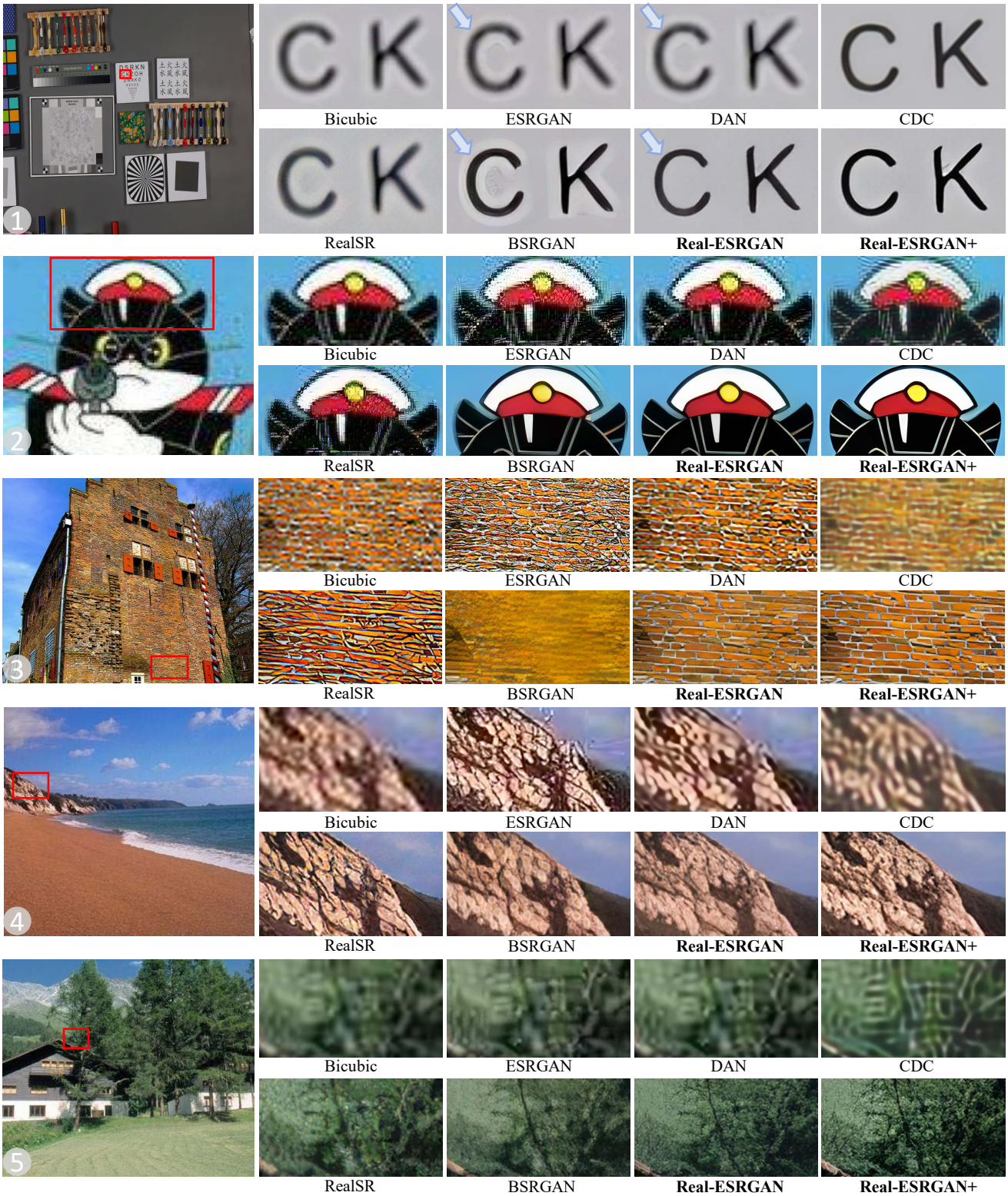

Figure 7: Qualitative comparisons on several representative real-world samples with upsampling scale factor of 4. Our Real-ESRGAN outperforms previous approaches in both removing artifacts and restoring texture details. Real-ESRGAN+ (trained with sharpened ground-truths) can further boost visual sharpness. Other methods may either fail to remove overshoot (the 1st sample) and complicated artifacts (the 2nd sample), or fail to restore realistic and natural textures for various scenes (the 3rd, 4th, 5th samples). (**Zoom in for best view**)

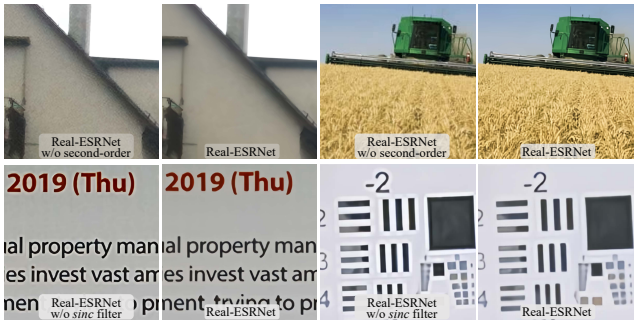

Figure 8: **Top**: Real-ESRNet results w/ and w/o second-order degradation process. **Bottom**: Real-ESRNet results w/ and w/o *sinc* filters. Zoom in for best view

4.3. Ablation Studies

Second-order degradation model. We conduct ablation studies of degradations on Real-ESRNet, as it is more controllable and can better reflect the influence of degradations. We replace the second-order process in Real-ESRNet with the classical degradation model to generate training pairs. As shown in Fig. 8 (Top), models trained with classical first-order degradation model cannot effectively remove noise on the wall or blur in the wheat field, while Real-ESRNet can handle these cases.

***sinc* filters**. If *sinc* filters are not employed during training, the restored results will amplify the ringing and overshoot artifacts that existed in the input images, as shown in Fig. 8 (Bottom), especially around the text and lines. In contrast, models trained with *sinc* filters can remove those artifacts.

U-Net discriminator with SN regularization. We first employ the ESRGAN setting including the VGG-style discriminator and its loss weights. However, we can observe from Fig. 9, this model cannot restore detailed textures (bricks and bushes) and even brings unpleasant artifacts in bush branches. Using a U-Net design could improve local details. Yet, it introduces unnatural textures and also increases training instability. SN regularization could improve restored textures while stabilizing training dynamics.

More complicated blur kernels. We remove the generalized Gaussian kernel and plateau-shaped kernel in blur synthesis. As shown in Fig. 10, on some real samples, the model cannot remove blur and recover sharp edges as Real-ESRGAN do. Nevertheless, on most samples, their differences are marginal, indicating that the widely-used Gaussian kernels with a high-order degradation process can already cover a large real blur space. As we can still observe slightly better performance, we adopt those more complicated blur kernels in Real-ESRGAN.

4.4. Limitations

Though Real-ESRGAN is able to restore most real-world images, it still has some limitations. As shown in Fig. 11, 1) some restored images (especially building and indoor scenes) have twisted lines due to aliasing issues. 2) GAN training introduces unpleasant artifacts on some samples. 3) It could not remove out-of-distribution complicated degradations in the real world. Even worse, it may amplify these artifacts. These drawbacks have great impact on the practical application of Real-ESRGAN, which are in urgent need to address in future works.

Figure 9: Ablation on the discriminator design. Zoom in for best view

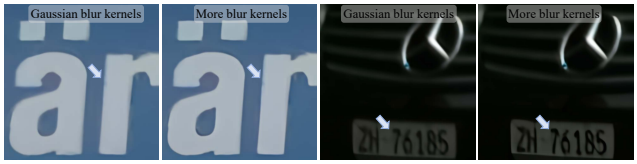

Figure 10: Ablation on using more blur kernels (generalized blur and plateau-shaped kernels). Zoom in for best view

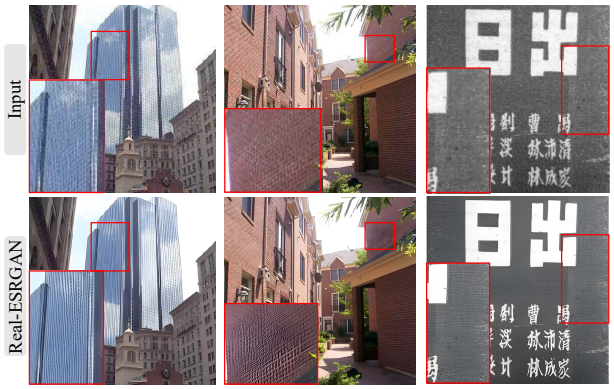

Figure 11: Limitations: 1) twisted lines; 2) unpleasant artifacts caused by GAN training; 3) unknown and out-of-distribution degradations. Zoom in for best view

5. Conclusion

In this paper, we train the practical Real-ESRGAN for real-world blind super-resolution with pure synthetic training pairs. In order to synthesize more practical degradations, we propose a high-order degradation process and employ *sinc* filters to model common ringing and overshoot artifacts. We also utilize a U-Net discriminator with spectral normalization regularization to increase discriminator capability and stabilize the training dynamics. Real-ESRGAN trained with synthetic data is able to enhance details while removing annoying artifacts for most real-world images.

Acknowledgement. This work is partially supported by National Natural Science Foundation of China (61906184), the Shanghai Committee of Science and Technology, China (Grant No. 21DZ1100800 and 21DZ1100100).

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPRW*, 2017. 6

[2] Sefi Bell-Kligler, Assaf Shocher, and Michal Irani. Blind super-resolution kernel estimation using an internal-gan. In *NeurIPS*, 2019. 1, 2

[3] Yochai Blau, Roey Mechrez, Radu Timofte, Tomer Michaeli, and Lihi Zelnik-Manor. The 2018 pirm challenge on perceptual image super-resolution. In *ECCVW*, 2018. 6, 12

[4] Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *CVPR*, 2018. 6

[5] Jianrui Cai, Hui Zeng, Hongwei Yong, Zisheng Cao, and Lei Zhang. Toward real-world single image super-resolution: A new benchmark and a new model. In *ICCV*, 2019. 2, 6, 12

[6] Kelvin C.K. Chan, Xintao Wang, Xiangyu Xu, Jinwei Gu, and Chen Change Loy. Glean: Generative latent bank for large-factor image super-resolution. In *CVPR*, 2021. 2

[7] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *CVPR*, 2019. 2

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 12

[9] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014. 1, 2

[10] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE TPAMI*, 38(2):295–307, 2016. 1, 2

[11] Michael Elad and Arie Feuer. Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images. *IEEE transactions on image processing*, 6(12):1646–1658, 1997. 1, 2, 3

[12] Manuel Fritsche, Shuhang Gu, and Radu Timofte. Frequency separation for real-world super-resolution. In *ICCVW*, 2019. 2

[13] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *ICCV*, 2009. 1

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 2, 6

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 2

[16] Jinjin Gu, Hannan Lu, Wangmeng Zuo, and Chao Dong. Blind super-resolution with iterative kernel correction. In *CVPR*, 2019. 2

[17] Muhammad Haris, Greg Shakhnarovich, and Norimichi Ukita. Deep backprojection networks for super-resolution. In *CVPR*, 2018. 2

[18] Andrey Ignatov, Nikolay Kobyshev, Radu Timofte, Kenneth Vanhoey, and Luc Van Gool. Dslr-quality photos on mobile devices with deep convolutional networks. In *ICCV*, 2017. 6, 12

[19] Xiaozhong Ji, Yun Cao, Ying Tai, Chengjie Wang, Jilin Li, and Feiyue Huang. Real-world super-resolution via kernel estimation and noise injection. In *CVPRW*, 2020. 1, 2, 6, 12

[20] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 1, 6

[21] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016. 1, 2

[22] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, 2016. 2

[23] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[24] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, 2017. 2

[25] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 1, 2

[26] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 2, 6

[27] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPRW*, 2017. 1, 2

[28] Anran Liu, Yihao Liu, Jinjin Gu, Yu Qiaoand, and Chao Dong. Blind image super-resolution: A survey and beyond. *arXiv:2107.03055*, 2021. 2

[29] Ce Liu and Deqing Sun. On bayesian adaptive video super resolution. *IEEE transactions on pattern analysis and machine intelligence*, 36(2):346–360, 2013. 1, 2, 3

[30] Ding Liu, Bihan Wen, Yuchen Fan, Chen Change Loy, and Thomas S Huang. Non-local recurrent network for image restoration. In *NeurIPS*, 2018. 2

[31] Yu-Qi Liu, Xin Du, Hui-Liang Shen, and Shu-Jie Chen. Estimating generalized gaussian blur kernels for out-of-focus image deblurring. *IEEE Transactions on circuits and systems for video technology*, 2020. 3, 11

[32] Michael R Lomnitz. Diffjpeg. https://github.com/mlomnitz/DiffJPEG, 2021. 3

[33] Andreas Lugmayr, Martin Danelljan, and Radu Timofte. Unsupervised learning for real-world super-resolution. In *ICCVW*, 2019. 2

[34] Zhengxiong Luo, Yan Huang, Shang Li, Liang Wang, and Tieniu Tan. Unfolding the alternating optimization for blind super resolution. In *NeurIPS*, 2020. 2, 6, 12

[35] Tomer Michaeli and Michal Irani. Nonparametric blind super-resolution. In *CVPR*, 2013. 1

[36] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a completely blind image quality analyzer. *IEEE Signal Process. Lett.*, 20(3):209–212, 2013. 12

[37] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018. 2, 5

[38] Seonghyeon Nam, Youngbae Hwang, Yasuyuki Matsushita, and Seon Joo Kim. A holistic approach to cross-channel image noise modeling and its application to image denoising. In *CVPR*, 2016. 3

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015. 2

[40] Mehdi SM Sajjadi, Bernhard Schölkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *ICCV*, 2017. 2

[41] Edgar Schonfeld, Bernt Schiele, and Anna Khoreva. A u-net based discriminator for generative adversarial networks. In *CVPR*, 2020. 2, 5

[42] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016. 5

[43] Richard Shin and Dawn Song. Jpeg-resistant adversarial images. In *NeurIPS Workshop on Machine Learning and Computer Security*, 2017. 3

[44] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *CVPR*, 2017. 2

[45] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, Lei Zhang, Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, Kyoung Mu Lee, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *CVPRW*, 2017. 2, 6

[46] Longguang Wang, Yingqian Wang, Xiaoyu Dong, Qingyu Xu, Jungang Yang, Wei An, and Yulan Guo. Unsupervised degradation representation learning for blind super-resolution. In *CVPR*, 2021. 2

[47] Xintao Wang, Yu Li, Honglun Zhang, and Ying Shan. Towards real-world blind face restoration with generative facial prior. In *CVPR*, 2021. 2

[48] Xintao Wang, Ke Yu, Kelvin C.K. Chan, Chao Dong, and Chen Change Loy. Basicsr. https://github.com/xinntao/BasicSR, 2020. 6

[49] Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Recovering realistic texture in image super-resolution by deep spatial feature transform. In *CVPR*, 2018. 2, 6, 12

[50] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *ECCVW*, 2018. 1, 2, 5, 6, 12

[51] Pengxu Wei, Ziwei Xie, Hannan Lu, ZongYuan Zhan, Qixiang Ye amd Wangmeng Zuo, and Liang Lin. Component divide-and-conquer for real-world image super-resolution. In *ECCV*, 2020. 2, 6, 12

[52] Yitong Yan, Chuangchuang Liu, Changyou Chen, Xianfang Sun, Longcun Jin, Peng Xinyi, and Xiang Zhou. Fine-grained attention and feature-sharing generative adversarial networks for single image super-resolution. *IEEE Transactions on Multimedia*, 2021. 2, 5

[53] Ke Yu, Xintao Wang, Chao Dong, Xiaoou Tang, and Chen Change Loy. Path-restore: Learning network path selection for image restoration. *arXiv:1904.10343*, 2019. 2

[54] Yuan Yuan, Siyuan Liu, Jiawei Zhang, Yongbing Zhang, Chao Dong, and Liang Lin. Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks. In *CVPRW*, 2018. 2

[55] Kai Zhang, Jingyun Liang, Luc Van Gool, and Radu Timofte. Designing a practical degradation model for deep blind image super-resolution. *arXiv preprint arXiv:2103.14006*, 2021. 2, 3, 4, 6, 12

[56] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Learning a single convolutional super-resolution network for multiple degradations. In *CVPR*, 2018. 1, 2

[57] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, 2018. 2

[58] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, 2018. 2

[59] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 2019. 6, 12

[60] Ruofan Zhou and Sabine Susstrunk. Kernel modeling super-resolution on real low-resolution images. In *ICCV*, 2019. 2

Appendix

A. Details of Classical Degradation Model

In this section, we provide more details (especially examples) of each degradation type used in the classical degradation model.

A.1. Blur

Isotropic and anisotropic Gaussian filters are the common choices for blur kernels. We show several Gaussian kernels and their corresponding blurry images in Fig. 12.

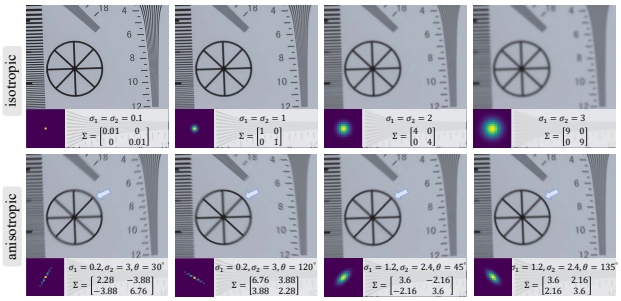

Figure 12: Examples of Gaussian kernels (kernel size 21) and their corresponding blurry images. Zoom in for best view

To include more diverse kernel shapes, we further adopt generalized Gaussian blur kernels [31] and a plateau-shaped distribution. Fig. 13 shows how the shape parameter β controls kernel shapes. Empirically, we found that including these blur kernels produces sharper outputs for several real samples.

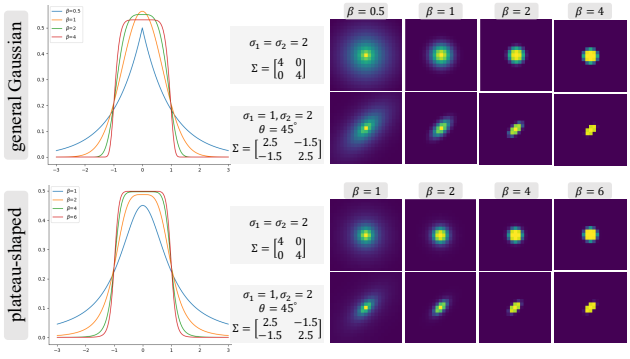

Figure 13: Blur kernels with different shape parameters in general Gaussian distribution and plateau-shaped distribution. Zoom in for best view

A.2. Noise

Fig. 14 depicts the additive Gaussian noise and Poisson noise. Poisson noise has an intensity proportional to the image intensity, and the noises at different pixels are independent of one another. As shown in Fig. 14, the Poisson noise has low noise intensity in dark areas.

Figure 14: Visual comparisons of Gaussian and Poisson noises. Poisson noise has low noise intensity in dark areas. Zoom in for best view

A.3. Resize

There are several resize algorithms. We compare the following resize operations: nearest-neighbor interpolation, area resize, bilinear interpolation and bicubic interpolation. We examine the different effects of these resize operations. We first downsample an image by a scale factor of four and then upsample to its original size. Different downsampling and upsampling algorithms are performed, and the results of different combinations are shown in Fig. 15. It is observed that different resize operations result in very different effects - some produce blurry results while some may output over-sharp images with overshoot artifacts.

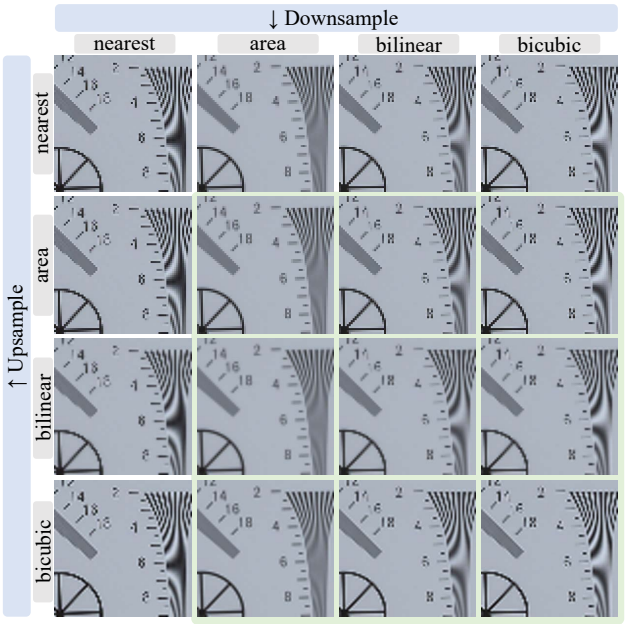

Figure 15: Effects of different combinations of down- and up-sampling algorithms. The images are first downsampled by a scale factor of four and then upsampled to its original size. Zoom in for best view

Table 1: NIQE scores on several diverse testing datasets with real-world images. The lower, the better.

	Bicubic	ESRGAN [50]	DAN [34]	RealSR [19]	CDC [51]	BSRGAN [55]	**Real-ESRGAN**	**Real-ESRGAN+**
RealSR-Canon [5]	6.1269	6.7715	6.5282	6.8692	6.1488	5.7489	4.5899	**4.5314**
RealSR-Nikon [5]	6.3607	6.7480	6.6063	6.7390	6.3265	5.9920	5.0753	**5.0247**
DRealSR [51]	6.5766	8.6335	7.0720	7.7213	6.6359	6.1362	4.9796	**4.8458**
DPED-iphone [18]	6.0121	5.7363	6.1414	5.5855	6.2738	5.9906	5.4352	**5.2631**
OST300 [49]	4.4440	3.5245	5.0232	4.5715	4.7441	4.1662	2.8659	**2.8191**
ImageNet val [8]	7.4985	**3.6474**	6.0932	3.8303	7.0441	4.3528	4.8580	4.6448
ADE20K val [59]	7.5239	3.6905	6.3839	**3.4102**	6.9219	3.9434	3.7886	3.5778

A.4. JPEG compression

We use the PyTorch implementation - `DiffJPEG`. We observe that the compressed images by `DiffJPEG` are a bit different from those compressed by the `cv2` package. Fig. 16 shows the typical JPEG compression artifacts and the difference caused by using different packages. Such a difference may bring an extra gap between synthetic and real samples. In this work, we only adopt `DiffJPEG` for simplicity, and this difference will be addressed later.

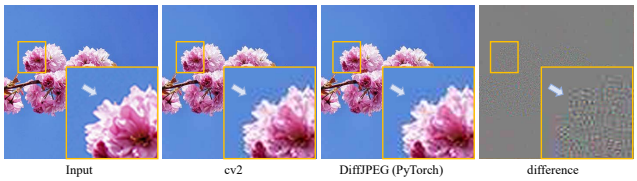

Figure 16: JPEG compressed images by `cv2` and `DiffJPEG`, with quality factor $q = 50$. They produces slightly different results. Zoom in for best view

B. Quantitative Comparisons

We provide the non-reference image quality assessment - NIQE [36] for reference. Note that existing metrics for perceptual quality cannot well reflect the actual human perceptual preferences on the fine-grained scale [3].

We compare our Real-ESRGAN with several state-of-the-art methods, including ESRGAN [50], DAN [34], CDC [51], RealSR [19] and BSRGAN [55]. We test on several diverse testing datasets with real-world images, including RealSR [5], DRealSR [51], OST300 [49], DPED [18], ImageNet validation [8] and ADE20K validation [59]. The results are shown in Tab. 1. Though our Real-ESRGAN+ does not optimize for NIQE scores, it sill produces lower NIQE scores on most testing datasets.

C. More Qualitative Comparisons

We show more qualitative comparisons with previous works. As shown in Fig. 17, our Real-ESRGAN outperforms previous approaches in both removing artifacts and restoring texture details. Real-ESRGAN+ (trained with sharpened ground-truths) can further boost visual sharpness. Other methods typically fail to remove complicated artifacts (the 1st sample) and overshoot artifacts (the 2nd, 3rd sample), or fail to restore realistic and natural textures for various scenes (the 4th, 5th samples).

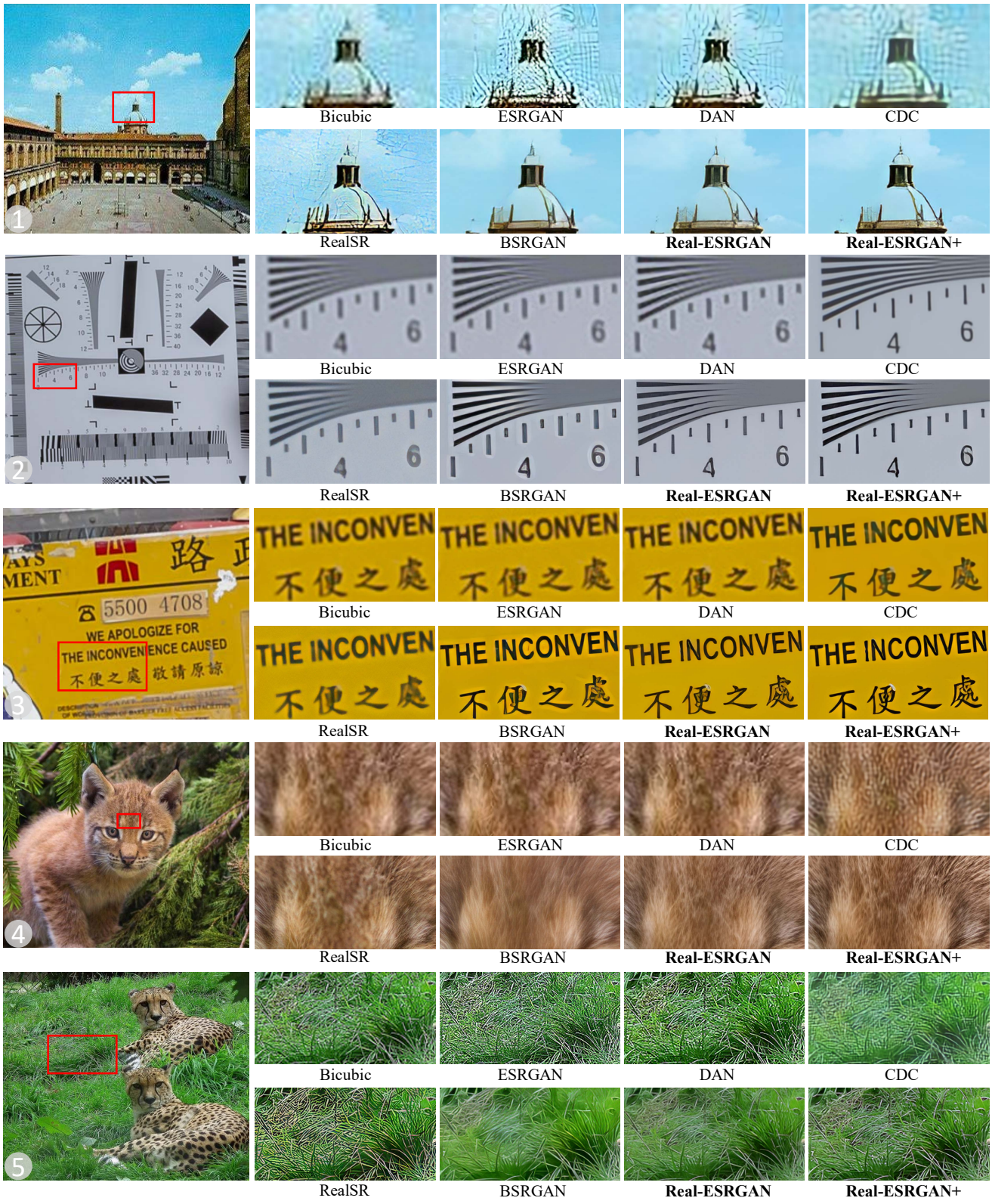

Figure 17: Qualitative comparisons on several representative real-world samples with upsampling scale factor of 4. Our Real-ESRGAN outperforms previous approaches in both removing artifacts and restoring texture details. Real-ESRGAN+ (trained with sharpened ground-truths) can further boost visual sharpness. Other methods typically fail to remove complicated artifacts (the 1st sample) and overshoot artifacts (the 2nd, 3rd sample), or fail to restore realistic and natural textures for various scenes (the 4th, 5th samples). (**Zoom in for best view**)