Frequency Separation for Real-World Super-Resolution

Manuel Fritsche Shuhang Gu Radu Timofte
Computer Vision Lab, ETH Zürich, Switzerland
{manuelf, shuhang.gu, radu.timofte}@ethz.ch

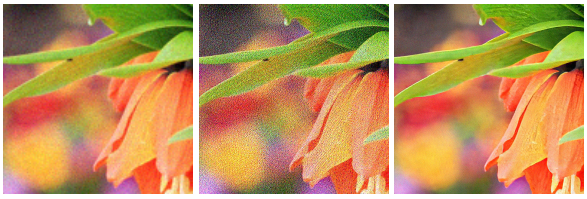
Abstract

Most of the recent literature on image super-resolution (SR) assumes the availability of training data in the form of paired low resolution (LR) and high resolution (HR) images or the knowledge of the downgrading operator (usually bicubic downscaling). While the proposed methods perform well on standard benchmarks, they often fail to produce convincing results in real-world settings. This is because real-world images can be subject to corruptions such as sensor noise, which are severely altered by bicubic downscaling. Therefore, the models never see a real-world image during training, which limits their generalization capabilities. Moreover, it is cumbersome to collect paired LR and HR images in the same source domain.

To address this problem, we propose DSGAN to introduce natural image characteristics in bicubically downscaled images. It can be trained in an unsupervised fashion on HR images, thereby generating LR images with the same characteristics as the original images. We then use the generated data to train a SR model, which greatly improves its performance on real-world images. Furthermore, we propose to separate the low and high image frequencies and treat them differently during training. Since the low frequencies are preserved by downsampling operations, we only require adversarial training to modify the high frequencies. This idea is applied to our DSGAN model as well as the SR model. We demonstrate the effectiveness of our method in several experiments through quantitative and qualitative analysis. Our solution is the winner of the AIM Challenge on Real World SR at ICCV 2019.

1. Introduction

The goal of image super-resolution (SR) is to increase the resolution in images. With the advent of convolutional neural networks (CNNs), the field has received increasing attention over the last couple of years. Modern techniques are now able to generate photo-realistic results on clean benchmark datasets. However, most state-of-the-art models [39, 36, 25] perform poorly on real-world images, which can be subject to corruptions such as sensor-noise. These characteristics usually lead to strange artifacts in the super-resolved images as shown in Figure 1.



original ESRGAN ours

Figure 1. $\times 4$ SR comparison of ESRGAN [36] and our method applied on a noisy input image. ESRGAN amplifies the corruptions, while our model preserves the noise level in the output.

The reason for this lies in the way these SR models are trained. Most of them rely on supervised training, which requires high resolution (HR) and corresponding low resolution (LR) image pairs. Since it is difficult to collect HR and LR images of the exact same scene, LR images are typically generated from HR images. In most cases, this is done by simply applying bicubic downscaling on HR images. While this method is easy and provides good results in clean settings, it comes with a significant problem: bicubic downscaling alters image characteristics. For example, it reduces corruptions in LR images, *i.e.* makes them "cleaner". Therefore, the model is only trained for input LR images that are altered by the downsampling operator. This leads to a significant performance drop when the model is applied on images that are not bicubically downscaled.

Since many real-world images have visible corruptions, the state-of-the-art SR methods are not very useful in practice. Current generations of smartphones are equipped with hardware that allows the deployment of powerful neural networks. Therefore, robust SR methods can be very useful for improving the quality of images taken from smartphone cameras. This work is focused on improving the performance of SR models in such real-world settings.

To achieve this, we aim to generate LR images that have the same characteristics as the images we want to super-resolve. These images allow SR models to be trained with a similar type of data that they encounter during application. In the first step, we create LR images by downsampling the

HR images with bicubic downscaling. In a second step, we alter the characteristics of these LR images to match those of the source images. This is done by using a Generative Adversarial Network (GAN) setup [10], which allows us to train a neural network to make our LR images indistinguishable from the source images. However, training a GAN is very difficult and needs to be stabilized to converge to the desired result. Similar to Ignatov *et al*. [12, 13], we achieve this stabilization by combining multiple loss functions: a color loss that forces the network to keep the low frequencies of the original image and an adversarial loss that uses the discriminator to produce high frequencies that look similar to the ones in the source images. Finally, we also add a perceptual loss that pushes the output towards solutions that look similar to the input images. Thereby, it ensures that the high frequencies generated by the GAN are still matching the low frequencies that are supervised by the color loss. This setup is based on the following idea: during the process of downsampling an image, the high frequencies are removed, while the low frequencies remain. Therefore, the resulting LR images lack the high frequency characteristics found in the original images. On the other hand, low image frequencies such as the color are preserved to an extend that depends on the downscaling factor. By limiting the adversarial loss to high frequencies, we greatly reduce the complexity of the task. This helps the discriminator to focus on the relevant image features and leaves others untouched. Therefore, our setup is more stable, converges faster and produces better results than a standard GAN.

Furthermore, we also apply our idea of separating the low and high image frequencies to train the SR model. Thereby, we use a similar strategy as mentioned above: use a pixel-wise loss to stabilize the low frequencies and apply the adversarial loss only on the high frequencies. Since this separates the pixel-wise and the adversarial loss, it simplifies the task of the discriminator. We also provide theoretical reasoning for why it makes sense to only train the high frequencies with a GAN and use a simple pixel-wise loss for the low frequencies.

We evaluate our methods on multiple datasets with artificial and natural corruptions[1]. To show the effectiveness of our implementation, we use the DF2K dataset that is a combination of the DIV2K [1, 33] and Flickr2K [25] datasets. Since this dataset contains clean images, it allows us to introduce artificial corruptions and create ground truth (GT) HR and LR image pairs by adding the same corruptions to both of them. We ran experiments with sensor noise as well as compression artifacts. In both cases we demonstrate the effectiveness of our methods through quantitative and qualitative evaluations. Furthermore, we ran our methods on real-world images from the DPED dataset [12], which were collected by an iPhone 3 camera. We only provide qualitative evaluation in this case since no GT is available. Finally, we also participated in the AIM 2019 Challenge on Real World Super-Resolution [27] associated with the AIM workshop at ICCV 2019. Our method won the first place in both tracks for source domain and target domain. None of our methods are specifically designed for a certain type of data. They can also be applied to images with other characteristics than the ones we used in our experiments.

[1]Our code and models are publicly available at `https://github.com/ManuelFritsche/real-world-sr`

2. Related Work

In recent years, the field of image super-resolution has been dominated by CNNs, which achieve state-of-the-art performance [33, 34, 4]. Dong *et al*. [6, 7] introduced the first top performing CNNs trained end-to-end to map LR to HR images. Based on this pioneering work, several refinements have been proposed [18, 17, 32, 21]. Thereby, deeper networks with residual layers such as EDSR [25] produce better results than standard CNNs. Additional improvements were made by using different variants of densely connected residual blocks [40, 36] as building blocks of the model. These blocks allow to further increase the depth of the networks, resulting in very powerful models.

Most of the previously mentioned methods are based on optimizing the L_1 or L_2 distance between the SR image and the ground truth HR image. While this strategy achieves state-of-the-art performance in image fidelity metrics such as PSNR, the resulting images are often blurry. This is because the human perception of visual similarity only has a limited correlation with such pixel-wise errors. Therefore, more recent SR methods are based on loss functions and training methods that are better suited to produce visually pleasing images. Gatys *et al*. [8, 9] show that high-level features extracted from pre-trained networks can be used to design perceptual loss functions. Such a loss function is used in [15] to enhance the visual quality of super-resolved images. The SRGAN model [22] is trained with an additional adversarial loss to push the output to the manifold of natural images, which allows to generate photo-realistic results. Several works have proposed further improvements with approaches that focus on perceptual similarity [29, 36, 39]. The recently introduced RankSRGAN [39] uses a method to train SR models on indifferentiable perceptual metrics. Our experiments are based on ESRGAN [36], the winner of the PIRM 2018 challenge on perceptual super-resolution [2]. It introduces several improvements to the SRGAN model, thereby achieving state-of-the-art perceptual performance.

All of the previously mentioned models are trained with HR/LR image pairs, generated through bicubic downscaling. Therefore, these models perform poorly in real-world scenarios. One way of addressing this issue is by directly collecting paired data, which is explored in recent work [4, 5]. However, these approaches rely on compli-

cated hardware and require the collection of new data for each camera source. Other methods try to make SR more robust, tailor them to the test image. Liang *et al*. [24] proposed to fine-tune a pre-trained SR model to the test image. ZSSR [30] is a lightweight CNN that is trained by only using the test image, which allows the network to focus only on image specific details. However, both approaches still rely on a known downsampling operation during training. Additionally, training a network for each test image results in very slow predictions. Yuan *et al*. [37] propose a model that learns a mapping from the original input to a clean input space, after which they apply super-resolution. They use a complex framework with two cycle-consistency losses, which increases training time. Their initial cleaning step improves the performance of the model on corrupted images, but also increases the complexity of their model. Conversely, our approach mainly focuses on generating training data. We only make small modifications in the discriminator and loss functions, which does not introduce more complexity in the model. Similar to our work, some novel methods generate paired data artificially. Kim *et al*. [16] propose an auto-encoder-based framework to jointly learn downsampling and upsampling. While their super-resolution method performs well on images downsampled by their model, it is not applicable to unknown downsampling operations. Bulat *et al*. [3] explore a method to learn the downsampling operation. However, they focus only on faces, but not the general super-resolution problem, which makes the task a lot easier. In contrast, we do not make any assumptions on the content of the images.

3. Proposed Method

3.1. Real-World Super-Resolution

State-of-the-art SR models rely on fully supervised training of neural networks on paired HR and LR images. While collecting images is not a difficult task, obtaining paired images from two different sources is difficult and cumbersome. For this reason, the SR field mainly relies on using bicubic downscaling to generate image pairs. Although this approach has helped the development of promising SR models, it is limiting the generalization to real-world images because it can drastically alter certain image characteristics such as sensor noise. An example of how this downsampling operation affects a real-world image can be seen in Figure 2.

For our analysis, we assume that we are given a set of *source* images, which have similar characteristics (*e.g*. the same sensor noise distribution). One can then define the HR images $\mathbf{y} \in \mathcal{Y}$ either directly as the source images or as modified versions thereof. Finally, the LR images $\mathbf{x} \in \mathcal{X}$ are generated by downsampling the HR images. As traditional downsampling operations alter the image characteristics, images from $\mathcal{X}$ differ from images in $\mathcal{Z}$. Since our goal is to upsample images from the domain $\mathcal{Z}$, we aim to have $\mathcal{X} = \mathcal{Z}$. In other words, the LR images seen in training should be indistinguishable from the source images. Therefore, we aim to map images from $\mathcal{X}$ to $\mathcal{Z}$, while preserving the image content.

Figure 2. This image is taken from the DPED dataset [12] and shows strong corruptions. When comparing a cropped region (middle) to the bicubically downscaled version of equal size (right), one can clearly see how this operation reduces corruptions.

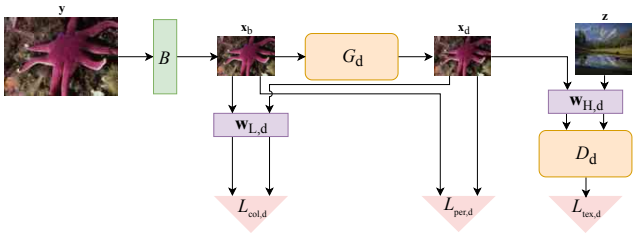


Figure 3. Visualizes the structure of the downsampling setup. B denotes the bicubic downscaling method, while the purple fields display the high- and low-pass filters. The red triangles denote the loss functions and the orange fields the neural networks.

3.2. Downsampling with Domain Translation

In the following, we describe a model that can produce realistic LR images in the source domain $\mathcal{Z}$, given HR images in some $\mathcal{Y}$ domain. The complete overall structure is shown in Figure 3.

In the first step, we bicubically downscale the HR image $\mathbf{y}$ to obtain a LR image $\mathbf{x}_\text{b} = B(\mathbf{y})$. Since $\mathbf{x}_\text{b}$ is now in the wrong domain $\mathcal{X}$, we use a neural network $G_\text{d}(\cdot)$ to translate it to the source domain $\mathcal{Z}$. We call this new image $\mathbf{x}_\text{d}$ with $\mathbf{x}_\text{d} = G_\text{d}(\mathbf{x}_\text{b})$. To train $G_\text{d}(\cdot)$ we use a standard GAN [10] setup with an additional discriminator $D_\text{d}(\cdot)$. The discriminator is trained with the generator output $G_\text{d}(B(\mathbf{y}))$ as fake data and uses the source images $\mathbf{z} \in \mathcal{Z}$ as real data.

Network Architectures The generator network consists mainly out of residual blocks [11] with two convolutional layers and ReLU activations in between. Except for the output layer, all convolutional layers use a 3×3 kernel with 64 features.

As image characteristics do not change the global image content, but only introduce local changes, we use a patch-

based discriminator [23, 14]. This discriminator is fully convolutional and returns a 2D array that determines which regions of the image look real or fake. The discriminator applies four convolutional layers with 5×5 kernels on the low-pass filtered input image. The number of output features of each convolutional layer increases from 64 to 128 and 256 with the final layer only using one feature map. Between the convolutional layers we apply Batch Normalization and LeakyReLU activations.

3.3. Frequency Separation

As described in Section 3.2, we are using a standard GAN setting to translate the original LR images to the source domain $\mathcal{Z}$. However, we do not just want any image in this source domain, but the one that is closest to our original LR image $\mathbf{x}_\text{b}$. One way of achieving this is by using multiple loss functions. By introducing a perceptual and a pixel loss, one can restrict the possible solutions that the generator produces. Unfortunately, such a loss function is hard to balance because we need the output of the generator $\mathbf{x}_\text{d}$ to stay close to the input $\mathbf{x}_\text{b}$ and at the same time introduce the image characteristics of the source domain. The result is that we have to deal with a trade-off and neither of the goals will be achieved perfectly.

This naive approach ignores the fact that we are dealing with downsampled images. As we discuss in Section 3.4, the downsampling process removes the high image frequencies and keeps the low frequency information within a reduced number of pixels. This leads to high frequency characteristics being lost, while low frequency information such as color and context remain. As the low image frequencies are preserved, we only need to alter the high frequencies in our mapping from $\mathcal{X}$ to $\mathcal{Z}$. Therefore, we propose to apply the discriminator only on the high frequencies $\mathbf{x}_\text{H,d}$ of $\mathbf{x}_\text{d}$ and keep the low frequencies $\mathbf{x}_\text{L,d}$ close to the original ones. This greatly reduces the complexity of the problem, making it easier for the discriminator to focus on the relevant image features. In our experiments, we found this method to be crucial in training the GAN. It not only speeds up the training process, but also produces better results. Furthermore, our GAN setup reduces undesired color shifts, because the discriminator ignores the low image frequencies.

We separate the low and high image frequencies by using simple linear filters. For this purpose, we first define a low-pass filter $\mathbf{w}_\text{L,d}$. The low and high frequencies can be obtained by simple convolutions:

$$\mathbf{x}_\text{L,d} = \mathbf{w}_\text{L,d} * \mathbf{x}_\text{d}, \tag{1}$$

$$\mathbf{x}_\text{H,d} = \mathbf{x}_\text{d} - \mathbf{x}_\text{L,d} = (\delta - \mathbf{w}_\text{L,d}) * \mathbf{x}_\text{d}. \tag{2}$$

Therefore, we define our high-pass filter as $\mathbf{w}_\text{H,d} = \delta - \mathbf{w}_\text{L,d}$. After applying a high-pass filter, we feed the remaining frequencies $\mathbf{x}_\text{H,d}$ to the discriminator $D_\text{d}(\cdot)$. The same high-pass filter is applied to the source images $\mathbf{z} \in \mathcal{Z}$. In our experiments, we empirically chose a moving average with kernel size 5 as low-pass filter. However, our method is not limited to any specific filter.

Loss Functions We combine multiple loss functions to train our model. The generator loss combines three different losses: color loss $L_\text{col,d}$, perceptual loss $L_\text{per,d}$ and texture loss $L_\text{tex,d}$, which is an adversarial loss.

The color loss is focusing on the low frequencies of the image. Since we do not want to change the low frequencies of $\mathbf{x}_\text{b}$, we apply an L_1 loss on these frequencies to keep them close to the input. The color loss is defined as

$$L_\text{col,d} = \frac{1}{m} \sum_{i=1}^{m} \left\| \mathbf{w}_\text{L,d} * G_\text{d}\left(\mathbf{x}_\text{b}^{(i)}\right) - \mathbf{w}_\text{L,d} * \mathbf{x}_\text{b}^{(i)} \right\|_1, \tag{3}$$

where m denotes the batch size.

As discussed in Section 3.3, we only apply the GAN loss on the high frequencies of the output $\mathbf{x}_\text{d}$, which results in the following loss for generator and discriminator:

$$L_\text{tex,d} = -\frac{1}{m} \sum_{i=1}^{m} \text{mean}\left(\log D_\text{d}\left(\mathbf{w}_\text{H,d} * G_\text{d}\left(\mathbf{x}_\text{b}^{(i)}\right)\right)\right), \tag{4}$$

$$\begin{aligned} L_{D_\text{d}} = -\frac{1}{m} \sum_{i=1}^{m} &\text{mean}\left(\log D_\text{d}\left(\mathbf{w}_\text{H,d} * \mathbf{z}^{(i)}\right)\right) \\ &+ \text{mean}\left(\log\left(1 - D_\text{d}\left(\mathbf{w}_\text{H,d} * G_\text{d}\left(\mathbf{x}_\text{b}^{(i)}\right)\right)\right)\right). \end{aligned} \tag{5}$$

Since the discriminator $D_\text{d}(\cdot)$ returns a 2D array of values and not just a single one, we take the mean over all these values in the loss function.

Finally, to ensure that the high and low frequencies fit together, we also apply a perceptual loss $L_\text{per,d}$ to $\mathbf{x}_\text{b}$ and $G_\text{d}(\mathbf{x}_\text{b})$. For this loss we use LPIPS [38], which is based on the features of the VGG network [31].

We define the complete loss functions of the generator as

$$L_{G_\text{d}} = L_\text{col,d} + 0.005 \cdot L_\text{tex,d} + 0.01 \cdot L_\text{per,d}. \tag{6}$$

3.4. Frequency Separation for Super-Resolution

We also apply our idea of frequency separation directly on ESRGAN [36]. However, the approach is not limited to this SR model and can easily be adapted for other methods. Our changes to the model are visualized in Figure 4.

For our analysis, we look at an image $\mathbf{y}$ that we downsample by a factor r. Let us assume that $\mathbf{x}$ is a downsampled version of $\mathbf{y}$ without aliasing. The Sampling Theorem tells us that $\mathbf{x}$ allows us to infer the lowest $1/r$ fraction of the possible frequencies of the original image $\mathbf{y}$, which we denote as $\mathbf{y}_\text{L}$ in the following. The remaining high frequencies are defined as $\mathbf{y}_\text{H} = \mathbf{y} - \mathbf{y}_\text{L}$. There is no need to consider the context and generate fake details to map from $\mathbf{x}$

to $\mathbf{y}_\mathrm{L}$. In contrast to the mapping from $\mathbf{x}$ to $\mathbf{y}$, the mapping from $\mathbf{x}$ to $\mathbf{y}_\mathrm{L}$ is a one-to-one mapping, which allows it to be reconstructed directly. For reconstructing $\mathbf{y}_\mathrm{H}$ on the other hand, we can only rely on the context information, because it contains all the high frequencies that are removed by the downsampling and anti-aliasing process. Thus, the mapping from $\mathbf{x}$ to $\mathbf{y}_\mathrm{L}$ is considerably easier to learn than the mapping from $\mathbf{x}$ to $\mathbf{y}_\mathrm{H}$.

Similar to Section 3.3, we use a low-pass filter $\mathbf{w}_\mathrm{L}$ and a high-pass filter $\mathbf{w}_\mathrm{H}$ to split up the low and high image frequencies. In our experiments, we found that a simple moving average with kernel size 9 works well. The low frequencies of $G(\mathbf{x})$ can be learned directly over the L_1 loss, since there is only one possible $\mathbf{y}_\mathrm{L}$ given $\mathbf{x}$. Since the colors of an image are mainly defined in the low frequencies, we call this loss the color loss L_col:

$$L_\mathrm{col} = \frac{1}{m}\sum_{i=1}^{m}\left\|\mathbf{w}_\mathrm{L} * G\left(\mathbf{x}^{(i)}\right) - \mathbf{w}_\mathrm{L} * \mathbf{y}^{(i)}\right\|_1. \quad (7)$$

The high frequencies of $G(\mathbf{x})$ on the other hand have multiple ground truth values and cannot be learned through a pixel-based loss. Therefore, we use the adversarial loss only on these high frequencies, by simply adding a high-pass filter in front of the discriminator. This greatly reduces the complexity of the task, as the discriminator does not have to deal with the low frequencies.

To make sure the high frequency details generated by the GAN loss match the low frequencies, the perceptual loss is applied on the full output. This results in the following adapted loss function for the ESRGAN generator:

$$L_G = L_\mathrm{per} + 0.005 \cdot L_\mathrm{adv} + 0.01 L_\mathrm{col}, \quad (8)$$

This loss function simplifies the task of the discriminator, which allows the model to produce outputs that match the target distribution more closely.

4. Experiments

4.1. Experimental Setup

Dataset Generation For all experiments, we use a scaling factor of 4 between the HR and LR images. We generate the HR/LR image pairs by using the model described in Section 3.1, which we call *DSGAN* (DownSampleGAN). We train it with 512×512 image patches, which we bicubically downscale with the MATLAB imresize method. For the discriminator, we use random 128×128 crops of the source images $\mathbf{z} \in \mathcal{Z}$. Using a batch size of 16 image patches, we train the model for 200 or 300 epochs, depending on the size of the dataset. Similar to CycleGAN [41], we use Adam optimizer [19] with $\beta_1 = 0.5$ and an initial learning rate of $2 \cdot 10^{-4}$. The learning rate is kept constant for the first half of the epochs and then linearly decayed to 0 during the remaining epochs.

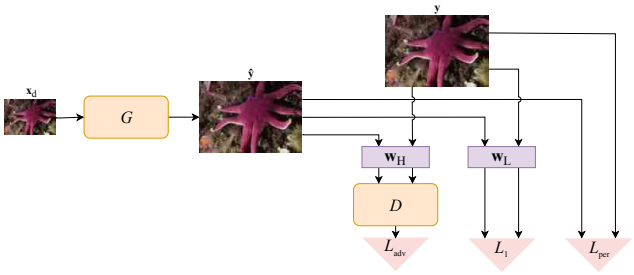


Figure 4. Visualizes our changes to the ESRGAN structure. Only the purple blocks are added to filter the images.

Same Domain Super-Resolution (SDSR): In this setup, we aim to generate a training dataset with HR and LR images that are both in the source domain $\mathcal{Y} = \mathcal{Z}$. We use the source images directly as the HR images and train our DSGAN model to map from the domain of bicubically downscaled LR images $\mathcal{X}$ to the domain of the HR images $\mathcal{Y}$.

Target Domain Super-Resolution (TDSR): If the images in the source domain have corruptions such as sensor noise, it is often desirable to remove these in the SR process. Therefore, in the TDSR setting, we aim to use HR images in a clean domain $\mathcal{Y} \neq \mathcal{Z}$ and only have our LR images in the source domain $\mathcal{Z}$. We generate the HR images by bicubically downscaling the source images with a factor of 2. Since we use a scaling factor of 4 between the HR and LR images, bicubic downscaling removes almost all corruptions. Therefore, we assume that the bicubically downscaled HR images from the SDSR setting and the TDSR setting to be approximately in the same domain $\mathcal{X}$. Thus, we apply the same model trained for SDSR in our TDSR setting to generate the LR images in the $\mathcal{Z}$ domain.

ESRGAN and ESRGAN-FS In the second step, we use the HR/LR image pairs to train our SR model with either ESRGAN or our modified ESRGAN described in Section 3.4, which we denote as *ESRGAN-FS* in the following. We initialize training with the fully pre-trained ESRGAN weights in both cases, as training ESRGAN from scratch takes a long time. We then perform $50k$ training iterations with an initial learning rate of 10^{-4}. The learning rate is halved after $5k$, $10k$, $20k$ and $30k$ iterations. We use Adam optimizer [19] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for both generator and discriminator training.

Datasets For our experiments with artificial corruptions, we use the DF2K dataset, which is a merger of the DIV2K [1, 33] and Flickr2K [25] datasets. It contains 3450 train images and 100 validation images with very little corruptions. To evaluate our method, we introduce two kinds of corruptions: sensor noise and compression artifacts. The sensor noise is modeled by adding pixel-wise independent

Gaussian noise with zero mean and a standard deviation of 8 pixels to the images. The compression artifacts are introduced by converting the images to JPEG with a quality of 30. Thereby, we create ground truth HR/LR pairs by applying the same degradation on both HR and LR images.

To test our methods on real-world data, we use images from the DPED dataset [12]. More precisely, we use the 5614 train and 113 test images taken with an iPhone 3 camera. Since we cannot generate any ground truth for these images, we only compare the results visually.

Furthermore, we also participated in the AIM 2019 Challenge on Real World Super-Resolution [27]. In this challenge 2650 corrupted source images and 800 clean target images are provided for training. The corruptions in the source data are artificial but unknown. The validation and test set contain 100 images each and have the same corruptions as the source data [26].

Quantitative Evaluation For our quantitative evaluation, we use the popular PSNR and SSIM methods, which we calculate with the scikit-image measure module [35]. While SSIM and PSNR are often used for measuring similarity to ground truth images, the resulting similarity values often correlate poorly with actual perceived similarity. Therefore, we also use LPIPS [38] for comparison. As mentioned in Section 3.3, this measure is based on the features of pre-trained neural networks, which have been shown empirically to correlate better with human perception than hand-crafted methods. We use the LPIPS method that is based on the features of AlexNet [20] for evaluation.

4.2. Comparison with State-of-the-Art

In this section, we compare our methods with four other state-of-the-art methods. The first one is ESRGAN [36], where we report the results with and without additional fine-tuning on the corrupted dataset (using bicubic downscaling). The fine-tuned model is referred to as ESRGAN (FT). We also look at another method called RankSRGAN [39]. This method uses an additional model called Ranker to simulate the behavior of indifferentiable perceptual metrics. It then trains the model with these simulated perceptual metrics. We use the pre-trained weights based on the NIQE metric [28]. Furthermore, we look at EDSR [25], which is a method optimized for PSNR-based super-resolution. We also include ZSSR [30] in our comparison, which applies a Zero-Shot learning strategy on each image it super-resolves.

Experiments on Corrupted Images In this experiment, our methods are fine-tuned on the corrupted DF2K datasets as described in Section 4.1. In Table 1, we compare the PSNR, SSIM and LPIPS values of all methods on the DIV2K validation set. As ground truth, we use the corrupted and original HR images in the SDSR and TDSR settings, respectively. In our discussion of the quantitative evaluation, we focus on the LPIPS measure, as it has the best correlation with image similarity. A qualitative comparison is shown in Figure 5.

All state-of-the-art methods introduce severe artifacts in the output for both sensor noise and compression artifacts. Specifically, ESRGAN achieves the worst LPIPS value in both the TDSR and SDSR settings. This value can be improved through fine-tuning on the corrupted images. However, the visual quality does not show any notable improvements. RankSRGAN achieves the best LPIPS value of the state-of-the-art methods, but still introduces significant corruptions in the output. EDSR and ZSSR manage to reduce the corruptions as they generally produce more blurry results. However, the LPIPS value is still close to the value for ESRGAN. In contrast, our methods produce satisfying

Figure 5. Qualitative comparison of our approaches (bold) with other state-of-the-art methods on images with additional sensor noise (upper two rows) or corruption artifacts (lower two rows). **ours** refers to our ESRGAN-FS trained with data generated by our DSGAN method in the two different settings.

		sensor noise			compression artifacts		
	method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SDSR	**ours**	20.87	0.39	0.26	22.03	0.57	0.35
	ESRGAN (FT)	18.32	0.22	0.64	23.29	0.62	0.42
	ESRGAN [36]	17.18	0.19	0.74	22.60	0.60	0.47
	RankSRGAN [39]	19.78	0.27	0.56	23.34	0.61	0.43
	EDSR [25]	23.40	0.44	0.71	23.96	0.64	0.45
	ZSSR [30]	23.18	0.43	0.73	24.02	0.64	0.46
TDSR	**ours**	22.52	0.52	0.33	20.39	0.50	0.42
	ESRGAN (FT)	18.61	0.24	0.81	23.10	0.60	0.50
	ESRGAN [36]	17.39	0.19	0.94	22.43	0.58	0.53
	RankSRGAN [39]	20.19	0.30	0.68	23.16	0.59	0.49
	EDSR [25]	24.48	0.53	0.68	23.75	0.62	0.54
	ZSSR [30]	24.21	0.52	0.70	23.81	0.62	0.55

Table 1. We report the mean scores over the DIV2K validation set with added corruptions. The arrows indicate if high ↑ or low ↓ values are desired. **ours** refers to our ESRGAN-FS trained with data generated by our DSGAN method.

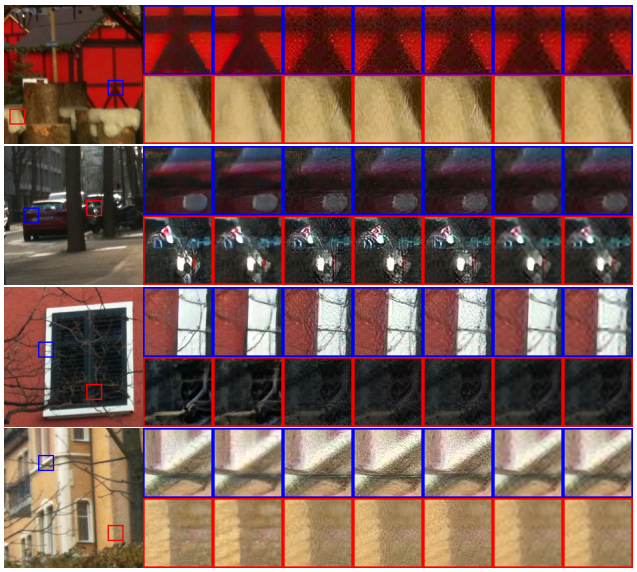


Figure 6. Qualitative comparison of our approaches (bold) with other state-of-the-art methods on real-world images from the DPED dataset taken with an iPhone 3 camera. **ours** refers to our ESRGAN-FS trained with data generated by our DSGAN method in the two different settings.

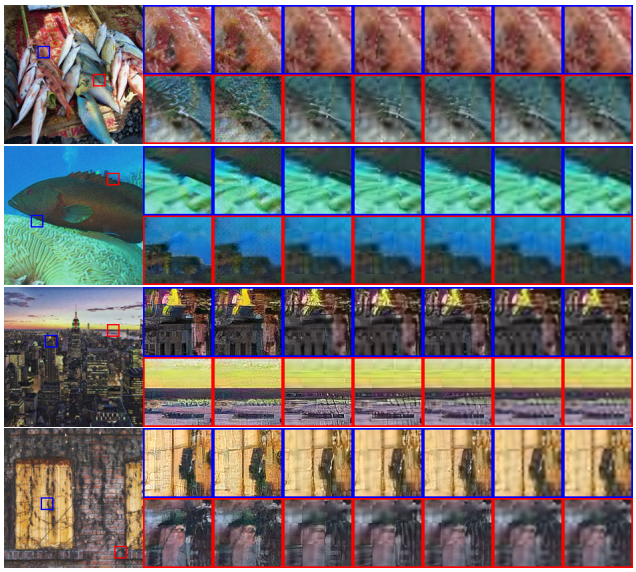


Figure 7. Qualitative comparison of our approaches (bold) with other state-of-the-art methods on the AIM2019 Real World SR test set. **ours** refers to our ESRGAN-FS trained with data generated by our DSGAN method in the two different settings.

results in all cases. This is reflected in the LPIPS values, as well as the visual quality of the images, which show almost no corruptions. This is due to our model being trained with input images with similar characteristics as the validation images that were upsampled to generate these results.

Experiments on Real-World Images We also evaluate our method on real-world images from the DPED dataset [12], where we use the train images captured by an iPhone 3 camera for fine-tuning. Since we do not have access to ground truth, we only compare the results visually, which is done in Figure 6. Due to the significant amount of corruptions (cf. Figure 2), none of the state-of-the-art methods produces satisfying results on this dataset. ESRGAN and RankSRGAN introduce strong artifacts, which are slightly reduced in the PSNR based EDSR. ZSSR produces visually very similar results, which are rather blurry. In contrast, our models produce images that are sharp and greatly reduce the amount of corruptions.

The AIM 2019 Challenge We also participated in the AIM 2019 Real World Super-Resolution Challenge [27]. Thereby, the SR ($\times 4$) images are supposed to either match the source domain in *Same Domain Real World Super-Resolution (SDSR)* or a clean target domain in *Target Domain Real World Super-Resolution (TDSR)*.

We first train our DSGAN model on the corrupted source dataset. To increase the diversity in the data, we randomly flip and rotate the images. For *SDSR*, our HR dataset contains the source and target datasets. Since the latter contains clean images, we use DSGAN to add the corruptions. In *TDSR*, the HR dataset is constructed by combining the target dataset and the bicubically downscaled source dataset, for which we use a scaling factor of 2. In both cases, the LR images are created by applying DSGAN on the HR images. We then fine-tune ESRGAN-FS on these datasets.

In Figure 7, we provide qualitative results for our SDSR and TDSR method and compare them to other state-of-the-art methods. As expected, PSNR based methods like EDSR [25] produce blurry results. Perception-driven methods, such as ESRGAN [36], generate sharper images, but they also increase the effect of image corruptions. On the other hand, our method produces sharp images with only few corruptions in both the SDSR and TDSR case. Most notably, the block structure corruptions caused by compression artifacts in the input image are removed in our models.

Furthermore, we present the results from the challenge in Table 2. In addition to PSNR, SSIM and LPIPS, a Mean-Opinion-Score (MOS) was conducted to evaluate how similar the results are to ground truth. For both SDSR and TDSR our method won the challenge by achieving the lowest MOS. More information on the evaluation and competing methods can be found in the challenge report [27].

4.3. Ablation Study

In this section, we compare different versions and combinations of our models. We evaluate our method in the

	Method	PSNR↑	SSIM↑	LPIPS↓	MOS↓
SDSR	**MadDemon (ours), *winner***	22.65	0.48	**0.36**	**2.22**
	IPCV_IITM	25.15	0.60	0.66	2.36
	Nam	**25.52**	**0.63**	0.65	2.46
	CVML	24.59	0.62	0.61	2.47
	ACVLab-NPUST	24.77	0.61	0.60	2.49
	SeeCout	25.30	0.61	0.74	2.50
	Image Specific NN for RWSR	24.31	0.60	0.69	2.56
TDSR	**MadDemon (ours), *winner***	20.72	0.52	0.40	**2.34**
	SeeCout	21.76	0.61	**0.38**	2.43
	IPCV_IITM	**22.37**	**0.62**	0.59	2.51
	Image Specific NN for RWSR	21.97	**0.62**	0.61	2.59

Table 2. This table reports the quantitative results from the AIM 2019 Challenge on Real World SR [27]. The arrows indicate if high ↑ or low ↓ values are desired.

same setting as used in Section 4.2. Although we also report PSNR and SSIM, we focus mostly on LPIPS in our discussion as it correlates best with perceptual similarity. As ground truth, we use the corrupted and clean HR images in the SDSR and TDSR settings, respectively. Our quantitative analysis is provided in Table 3. Furthermore, we provide visual results on the DIV2K [1, 33] validation set in Figure 8 for sensor noise and compression artifacts.

We vary two things in our experiments: the model that is used for SR and the method that is used to generate the HR/LR image pairs. We compare ESRGAN [36] and ESRGAN-FS (with frequency separation) as our SR models. In each case, one of these models is fine-tuned with one of the following datasets.

bicubic: This standard method results in very poor SR performance. In all cases, it produces strong corruptions in the SR images. This is also reflected in the LPIPS values, which are the worst of all compared methods.

DSGAN: In all cases, using DSGAN greatly improves the performance of the method compared to using bicubic downscaling. Thereby, ESRGAN-FS tends to produce slightly sharper images than ESRGAN, which leads to a slightly worse LPIPS score for ESRGAN-FS, as the introduced details are often different from ground truth. Furthermore, ESRGAN-FS better matches the source characteristics in the output, which can be seen in the SDSR setting with compression artifacts. ESRGAN does not always manage to produce realistic artifacts, while the artifacts generated by ESRGAN-FS look convincing in all image regions.

GT: In case of input images with sensor noise, this results in images that look very similar to the ones that were generated by using DSGAN. This means that our DSGAN method is very accurate in reproducing the image characteristics of the source images. In case of compression artifacts, the difference is more apparent, as the models trained with the DSGAN dataset introduce some additional corruptions. Interestingly, the models trained with DSGAN produce sharper results than the models trained with GT. This might be because we use bicubic downscaling to generate these image pairs, which not only alters corruptions but also other image characteristics.

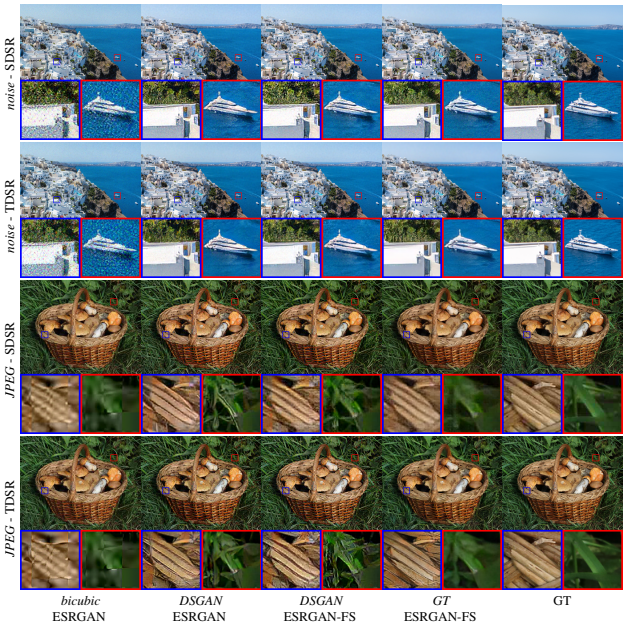


Figure 8. Ablation study on the DIV2K validation set with sensor noise (*noise*) or compression artifacts (*JPEG*). The first line below the images denotes the *dataset* used for fine-tuning, while the second line refers to the SR method.

			sensor noise			compression artifacts		
	dataset	SR method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SDSR	bicubic	ESRGAN	18.32	0.22	0.63	23.29	0.62	0.42
	DSGAN	ESRGAN	21.36	0.38	0.23	21.88	0.57	0.35
	DSGAN	ESRGAN-FS	20.87	0.39	0.26	22.03	0.57	0.35
	GT	ESRGAN-FS	22.49	0.41	0.18	23.26	0.62	0.35
TDSR	bicubic	ESRGAN	18.80	0.24	0.80	22.65	0.58	0.52
	DSGAN	ESRGAN	22.45	0.54	0.32	21.21	0.54	0.40
	DSGAN	ESRGAN-FS	22.52	0.52	0.33	20.39	0.50	0.42
	GT	ESRGAN-FS	25.02	0.69	0.21	22.43	0.59	0.34

Table 3. This table reports the quantitative results of our ablation study. The arrows indicate if high ↑ or low ↓ values are desired.

5. Conclusion

We propose DSGAN to generate paired HR and LR images with similar characteristics. We argue that the relevant characteristics mainly appear in the high frequencies of an image, which we exploit by applying the adversarial loss only on these frequencies. Furthermore, we also apply our idea of frequency separation to the SR model, which allows it to match the target distribution more closely. Our multiple experiments with artificial and natural corruptions demonstrate the effectiveness of our approach for real-world SR. We not only beat the state-of-the-art methods in these experiments, but also won the AIM 2019 Challenge on Real World Super-Resolution [27].

Acknowledgments. This work was partly supported by ETH General Fund and by Nvidia through a GPU grant.

References

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

ceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 126–135, 2017.

[2] Yochai Blau, Roey Mechrez, Radu Timofte, Tomer Michaeli, and Lihi Zelnik-Manor. The 2018 pirm challenge on perceptual image super-resolution. In *The European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[3] Adrian Bulat, Jing Yang, and Georgios Tzimiropoulos. To learn image super-resolution, use a GAN to learn how to do image degradation first. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VI*, pages 187–202, 2018.

[4] Jianrui Cai, Shuhang Gu, Radu Timofte, and Lei Zhang. Ntire 2019 challenge on real image super-resolution: Methods and results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[5] Chang Chen, Zhiwei Xiong, Xinmei Tian, Zheng-Jun Zha, and Feng Wu. Camera lens super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1652–1660, 2019.

[6] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV*, pages 184–199, 2014.

[7] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):295–307, 2016.

[8] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in neural information processing systems*, pages 262–270, 2015.

[9] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Andrey Ignatov, Nikolay Kobyshev, Radu Timofte, Kenneth Vanhoey, and Luc Van Gool. Dslr-quality photos on mobile devices with deep convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3277–3285, 2017.

[13] Andrey Ignatov, Nikolay Kobyshev, Radu Timofte, Kenneth Vanhoey, and Luc Van Gool. Wespe: Weakly supervised photo enhancer for digital cameras. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[15] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II*, pages 694–711, 2016.

[16] Heewon Kim, Myungsub Choi, Bee Lim, and Kyoung Mu Lee. Task-aware image downscaling. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IV*, pages 419–434, 2018.

[17] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 1646–1654, 2016.

[18] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 1637–1645, 2016.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[21] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5835–5843, 2017.

[22] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 105–114, 2017.

[23] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pages 702–716. Springer, 2016.

[24] Yudong Liang, Radu Timofte, Jinjun Wang, Yihong Gong, and Nanning Zheng. Single image super resolution - when model adaptation matters. *CoRR*, abs/1703.10889, 2017.

[25] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1132–1140, 2017.

[26] Andreas Lugmayr, Martin Danelljan, and Radu Timofte. Unsupervised learning for real-world super-resolution. *ICCV Workshops*, 2019.

[27] Andreas Lugmayr, Martin Danelljan, Radu Timofte, et al. Aim 2019 challenge on real-world image super-resolution: Methods and results. *ICCV Workshops*, 2019.

[28] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a completely blind image quality analyzer. *IEEE Signal Processing Letters*, 20(3):209–212, 2012.

[29] Mehdi S. M. Sajjadi, Bernhard Schölkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 4501–4510, 2017.

[30] Assaf Shocher, Nadav Cohen, and Michal Irani. "zero-shot" super-resolution using deep internal learning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3118–3126, 2018.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[32] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2790–2798, 2017.

[33] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. Ntire 2017 challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 114–125, 2017.

[34] Radu Timofte, Shuhang Gu, Jiqing Wu, and Luc Van Gool. Ntire 2018 challenge on single image super-resolution: Methods and results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[35] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[36] 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 *Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part V*, pages 63–79, 2018.

[37] 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 *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 701–710, 2018.

[38] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

[39] Wenlong Zhang, Yihao Liu, Chao Dong, and Yu Qiao. Ranksrgan: Generative adversarial networks with ranker for image super-resolution. *arXiv preprint arXiv:1908.06382*, 2019.

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

[41] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.