

# Tackling the Ill-Posedness of Super-Resolution through Adaptive Target Generation

Younghyun Jo<sup>1</sup>Seoung Wug Oh<sup>2</sup>Peter Vajda<sup>3</sup>Seon Joo Kim<sup>1</sup><sup>1</sup>Yonsei University<sup>2</sup>Adobe Research<sup>3</sup>Facebook

## Abstract

By the one-to-many nature of the super-resolution (SR) problem, a single low-resolution (LR) image can be mapped to many high-resolution (HR) images. However, learning based SR algorithms are trained to map an LR image to the corresponding ground truth (GT) HR image in the training dataset. The training loss will increase and penalize the algorithm when the output does not exactly match the GT target, even when the outputs are mathematically valid candidates according to the SR framework. This becomes more problematic for the blind SR, as diverse unknown blur kernels exacerbate the ill-posedness of the problem. To this end, we propose a fundamentally different approach for the SR by introducing the concept of the adaptive target. The adaptive target is generated from the original GT target by a transformation to match the output of the SR network. The adaptive target provides an effective way for the SR algorithm to deal with the ill-posed nature of the SR, by providing the algorithm with the flexibility of accepting a variety of valid solutions. Experimental results show the effectiveness of our algorithm, especially for improving the perceptual quality of HR outputs.

## 1. Introduction

The goal of image super-resolution (SR) is to generate a high-resolution (HR) image from its corresponding low-resolution (LR) counterpart. To solve this problem, previous state-of-the-art SR methods have focused on finding the underlying relationship between LR and HR patch pairs in natural images. Some methods exploited internally found LR-HR patch pairs across scales within a given image [6, 12], and other methods learned the mapping functions with the patch pairs from large external images as external example-based approaches [5, 41, 38].

For most learning-based approaches, training example pairs are simulated from a large collection of HR images in a self-supervised manner. Specifically, a LR image  $I_{LR}$  is simulated from a ground truth (GT) HR image  $I_{GT}$  as

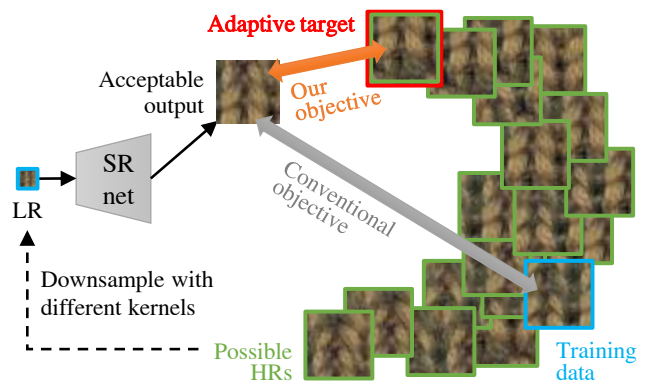


Figure 1. In (blind) SR, many HR patches can result in the same LR patch. Restricting the solution of an SR network to just one from the given ground truth can penalize a totally acceptable output. To tackle this issue, we propose an adaptive target strategy that generates new targets in order to relax the restriction on the possible solutions.

follows:

$$I_{LR} = (I_{GT} * k) \downarrow_s, \quad (1)$$

where  $k$  is a blur kernel,  $*$  is the convolution operator, and  $\downarrow_s$  is the downsampling operator with the scale factor  $s$ . The training set  $\mathcal{D} = \{(x_i, y_i)\}$  consists of the LR input patch  $x_i$  from  $I_{LR}$  and the HR target patch  $y_i$  from  $I_{GT}$  for the  $i$ -th example. Generally, an SR network  $f$  is trained to generate HR output  $f(x_i)$  close to the given target  $y_i$  by minimizing a loss function as follows:

$$\sum_i \ell(y_i, f(x_i)), \quad (2)$$

where  $\ell$  is a pixel reconstruction loss.

The SR problem is under-determined as different HR images can be mapped to the same LR image (Fig. 1). In other words, the given target  $y_i$  is not the only solution for the input  $x_i$ , as one input LR image can be originated from a variety of HR images. This becomes more significant when it comes to the blind SR task where the blur kernel  $k$  is unknown and diverse. A variety of unknown kernels makes the

relationship between inputs and targets inconsistent, such as differences in blurriness and pixel shift. As a result, the ill-posedness of the problem becomes exacerbated, and it is hardly solvable with conventional SR approaches without additional constraints. To cope with this, previous blind SR methods proposed specialized networks that first estimate the blur kernel and then take advantage of the estimated kernel information as an additional input for the SR network [8, 49, 2]. As expected, they are inherently sensitive to inaccurate kernel predictions [44].

In this paper, we propose a fundamentally different way to evaluate the output of an SR network by allowing more flexibility on the outputs of the algorithm. As can be seen in Fig. 1, many HR patches can be mapped to the same LR patch when downsampled with different kernels. Even if an SR network has generated an output that is close to one of the possible HR targets (green boxes), the conventional loss (Eq. (2)) will increase and penalize the SR network when only considering the corresponding HR target in the training data as the GT (blue box). This contrasts with the one-to-many mapping nature of the SR task.

To this end, we introduce a simple and effective way to encourage sharp output generation by accepting solutions other than the one from the training pair. Instead of directly comparing to the original target  $y_i$ , our method computes the loss for a new adaptive target  $\tilde{y}_i$  (red box). Conceptually, using our alternative target relaxes the typical pixel reconstruction loss by allowing various HR predictions given an LR input. The adaptive target is made from the original target to give the lowest penalty for the current network prediction  $f(x_i)$  while keeping the original contents and perceptual impression unchanged. Specifically, we find an affine transform matrix for every small non-overlapping piece of  $y_i$  to those of  $f(x_i)$  within the range of acceptable transforms. Then, each of the pieces is transformed to construct the adaptive target (Fig. 2). This process is conducted during the training on-the-fly with little computational overhead. For each training iteration, the SR network is trained using the loss computed with the adapted target.

Different from the previous blind SR methods that use additional kernel information, we present a novel way to learn blind SR by allowing diverse network outputs rather than constraining the solution space with the additional information. The proposed adaptive loss fully embraces the inconsistent nature of the blind SR by design, without the kernel estimation. In a way, we are taking advantage of the ill-posedness to generate sharper images, and there have been few investigations on whether the one-to-one paired training sample is adequate for the SR task [9, 23]. Our approach outperforms previous blind SR models by a good margin in terms of peak signal-to-noise ratio (PSNR) and visual quality. In addition, we found that our loss is also effective for non-blind SR especially when combined with

adversarial training (GAN) [7] for realistic SR.

In summary, the contributions of this paper are:

- We introduce a simple and effective way to encourage sharp output generation using proposed adaptive target as a solution for the one-to-many problem of SR. For blind SR, our method is fundamentally different from previous works as our algorithm works in a single-shot manner without the blur kernel estimation.
- The adaptive target is created on-the-fly during the training stage with little computational overhead, therefore, it is applicable to any training dataset without preprocessing. In addition, our framework can be attached to any deep SR models as a loss function.
- Our method outperforms previous state-of-the-art blind SR methods in terms of PSNR and visual quality. In non-blind scenario, our method generates consistent details when combined with GANs for perceptual SR, achieving both higher PSNR and LPIPS metrics at the same time.

## 2. Related Work

**Learning-Based Super-Resolution** The ability of deep neural networks (DNN) for solving the SR problem was first demonstrated by SRCNN [3], which consists of three convolutional layers. Most DNN based methods have improved the performance in terms of PSNR by stacking more convolutional layers and designing complex operation blocks or connections [15, 17, 22, 11, 48, 47, 21, 31, 27, 28]. The majority of previous works assume bicubic downsampling.

A number of studies have targeted to improve the perceptual quality of SR by adopting GANs [7] as an unsupervised loss for generating realistic details [18, 33, 30, 39, 37, 46, 32, 24]. Our approach complements GAN based methods very well by using the proposed adaptive target as supervision. We verify the performance gain by using our method together with GANs in Sec. 4.2.

**Blind Super-Resolution** In another line of study, DNN based blind SR methods have been proposed to cope with unknown downsampling kernels [8, 49, 2]. The blind SR methods employ a two-step approach: 1) blur kernel estimation and 2) non-blind SR using the kernel. As the first step, they focus on finding accurate kernels using the GT kernel for supervision as inaccurate kernels lead to artifacts in the final SR results. This is usually done by designing suitable deep network structures and loss functions for the kernel estimation. In the next stage, the estimated kernel and LR image are used together for SR.

In addition, zero-shot learning based SR methods can be applied to blind SR. ZSSR [34] trains a small deep SR network using a single test image to exploit the image-specific internal information. However, it takes time to update the network with a number of backpropagations. Recently proposed meta-learning [4] based SR methods can also be applied to blind SR [36, 29]. They train transferable param-

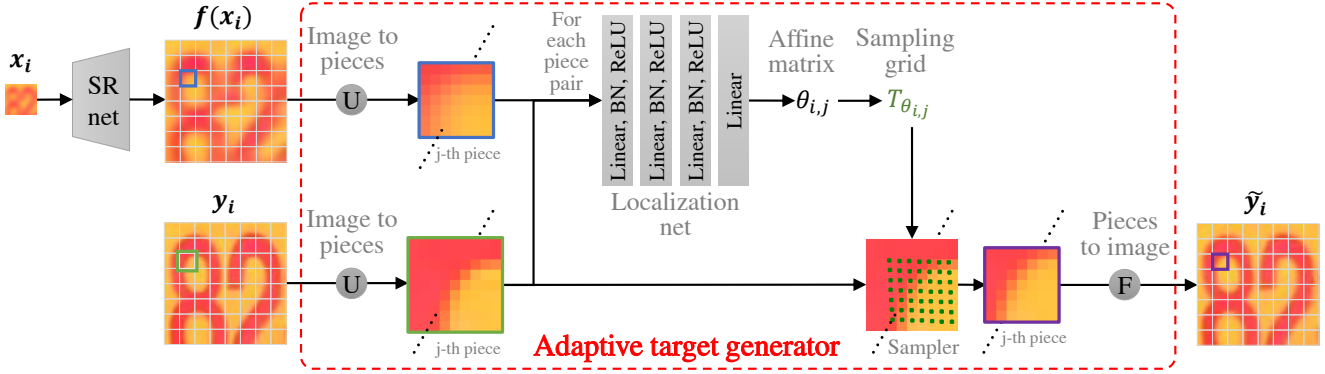


Figure 2. The overall process of generating our adaptive target. We transform every non-overlapping piece of an original target  $y_i$  (green box) to the corresponding area of the network output  $f(x_i)$  (blue box), and the transformed areas (purple box) are rearranged to generate our adaptive target  $\tilde{y}_i$ . Then, the training loss for the SR network  $f$  is computed between  $f(x_i)$  and  $\tilde{y}_i$ . The new target is designed to give less penalty for the acceptable areas.

ters of an SR network that can quickly adapt to any test image with a few gradient updates. Both the zero-shot learning and the meta-learning based methods bring new insights for the SR problem, however, the blur kernel needs to be known a priori for the adaptation at test time.

**Multiple Choice Learning** Our method is related to multiple choice learning (MCL) in the aspect of learning method. The basic idea of MCL is to learn to produce diverse outputs from a single model or ensemble of models and choose the best one among them [10, 19, 20]. For example, Li, Chen, and Koltun [20] trained an object segmentation model  $f$  that generates  $M$  diverse predictions  $\langle f_1(x_i), f_2(x_i), \dots, f_M(x_i) \rangle$ . With the MCL objective, every prediction is compared with a single solution as the target, and the prediction with the minimum loss is used for backpropagation. The model is encouraged to produce diverse outputs through the MCL loss, which is expressed as follows:

$$\sum_i \min_m \ell(y_i, f_m(x_i)). \quad (3)$$

Compared to MCL, our learning objective described in Sec. 3 has exactly the opposite purpose. In other words, we generate a single prediction while having multiple possible solutions as targets. This can be viewed as an approach for modeling one-to-many mapping such as SR. With our objective, the model focuses on giving only one correct answer, even if there are multiple possible answers.

### 3. Method

The overview of our method is depicted in Fig. 2. Our method consists of an SR network  $f$  and a proposed adaptive target generator (ATG). First, we pretrain  $f$  using a pixel reconstruction loss (e.g. Eq. (2)) with the original target  $y_i$  from a training set  $\mathcal{D} = \{(x_i, y_i)\}$ . Then,  $f$  is further trained with our adaptive target  $\tilde{y}_i$  generated from the ATG.

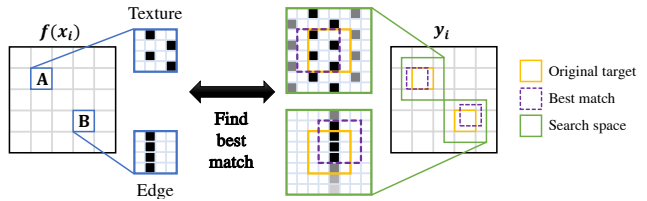


Figure 3. A simplified example for finding an adaptive target by only considering translational transformation. For every piece of  $f(x_i)$  (blue box), we search for the best match (purple box) in the search space (green box). Note that we assume both targets (original and best match) are mapped to the same LR patch when downsampled.

We consider the ideal adaptive target  $\hat{y}_i^*$  is the closest HR image to the SR result  $f(x_i)$  among possible HR images (Fig. 1). Formally, it can be expressed as follows:

$$\hat{y}_i^* = \operatorname{argmin}_{\hat{y}_i} \ell(\hat{y}_i, f(x_i)), \quad s.t. \quad x_i = (\hat{y}_i * k_i^m) \downarrow_s, \quad (4)$$

where  $\hat{y}_i$  is one of the possible HR images, thus it should become  $x_i$  when downsampled with a certain blur kernel  $k_i^m$ . However, it is mathematically very difficult to solve for  $\hat{y}_i^*$  as there exist an infinite number of  $\hat{y}_i$  and  $k_i^m$  pairs.

Therefore, we propose to find an approximated adaptive target  $\tilde{y}_i$ . In Fig. 3, we demonstrate the concept of the proposed adaptive target by only using the translation of pieces to build  $\tilde{y}_i$ . For every piece of  $f(x_i)$ , we set the search space around the corresponding target piece in  $y_i$ . The search space should not be too large because the original content needs to be maintained. We traverse the search space to find the area with closest match to the current output. The found areas are then gathered and rearranged to form  $\tilde{y}_i$ .

To allow more flexibility in generating adaptive targets, we generalize the above idea by considering affine trans-

| Layers       | Size of input channel   | Output channel          |
|--------------|-------------------------|-------------------------|
| FC-BN1d-ReLU | $C \times (p^2 + s^2)$  | $2C \times (p^2 + s^2)$ |
| FC-BN1d-ReLU | $2C \times (p^2 + s^2)$ | $2C \times (p^2 + s^2)$ |
| FC-BN1d-ReLU | $2C \times (p^2 + s^2)$ | $C \times (p^2 + s^2)$  |
| FC           | $C \times (p^2 + s^2)$  | 6                       |

Table 1. Network structures of the localization network in ATG.  $C$  is the number of channels of the input image, and  $p$  and  $s$  are the sizes of each piece for the SR network output  $f(x_i)$  and the original target  $y_i$  respectively.

formation in this paper. We propose a small neural network called ATG inspired by the spatial transformer network (STN) [13]. ATG transforms  $y_i$  toward  $f(x_i)$  through an affine transformation within acceptable distortion range. With ATG, we can approximate  $\hat{y}_i^*$  in a feed-forward manner without investigating all possible HRs (in Eq. (4)). Formally, our adaptive target is generated as follows:

$$\tilde{y}_i = \text{ATG}(y_i, f(x_i)). \quad (5)$$

Learning conventional SR networks with our adaptive target  $\tilde{y}_i$  instead of the given target  $y_i$  makes the networks produce sharper outputs for SR tasks.

### 3.1. Adaptive Target Generator (ATG)

The network design of ATG is depicted in Fig. 2. ATG have a localization network that consists of 4 fully connected layers, each followed by batch normalization and ReLU activation except for the last layer (Table 1). The localization network estimates an affine transformation matrix  $\theta_i$  that deforms the original target  $y_i$  to align it with the current SR network prediction  $f(x_i)$ . Through the transformation, our new target  $\tilde{y}_i$  with the minimal error compared to  $f(x_i)$  is created while keeping the original contents.

The process in ATG works in a patch-wise manner, not for the entire image. Specifically, the SR network’s output  $f(x_i)$  is divided into non-overlapping pieces of size  $p \times p$  with stride  $p$ , and the original target  $y_i$  is divided into overlapping pieces of size  $s \times s$  with the same stride  $p$  ( $p < s$ ). The pieces of  $y_i$  have a slightly larger size as the search space. In our experiments, we empirically set  $p = 7$  and  $s = 9$ . The piece pairs are fed into the localization network, and the localization network estimates affine transformation matrices  $\theta_{i,j}$  for transforming every  $j$ -th piece of  $y_i$  to the corresponding piece of  $f(x_i)$ . To transform each piece of  $y_i$ , the sampling grid (green dots in Fig. 2) is generated from the  $\theta_{i,j}$  and the bilinear sampling is applied for the transformation (please refer to the details in [13]). All the transformed pieces of size  $p \times p$  are then combined to generate our adaptive target  $\tilde{y}_i$ .

### 3.2. Learning with Adaptive Target

**Pretraining ATG** For accurate affine transformation matrix estimation, we pretrain the localization network in ATG in advance by using a synthetic affine matrix. In this way, we can provide direct supervision on the resulting affine matrix. We randomly simulate an affine matrix  $\theta^{Syn}$  combining two basic transformations, translation and rotation, in random order. Specifically, small amounts of random translation and random rotation are performed within range  $[-1, 1]$  pixel and  $[-10, 10]$  degree respectively to approximately satisfy the condition in Eq. (4), which is  $x_i \approx (\tilde{y}_i * k_i^n)$  for a certain blur kernel  $k_i^n$ . Then, we simulate the transformed image  $\tilde{y}_i^{Syn}$  from  $y_i$  by using  $\theta_{i,j}^{Syn}$ . We have tested with different combinations of the basic transforms including translation, rotation, scale, and shear, but simply using translation and rotation shows the best results as validated in Sec. 4.3.

Now, we have the original image  $y_i$ , the randomly distorted image  $\tilde{y}_i^{Syn}$ , and the corresponding transformation matrices  $\theta_{i,j}^{Syn}$ . The ATG gets  $y_i$  and  $\tilde{y}_i^{Syn}$  as input and estimates affine matrices  $\theta_{i,j}$ . Then, it is pretrained using losses on the affine matrix and the transformed image as follows:

$$\sum_i \left[ \sum_j \ell(\theta_{i,j}, \theta_{i,j}^{Syn}) + \lambda \ell(\tilde{y}_i, \tilde{y}_i^{Syn}) \right], \quad (6)$$

where  $\ell$  is the mean square error (MSE) and  $\lambda$  is a scaling parameter that we set to 0.01.

**Training SR Net with ATG** Finally, the pretrained SR network  $f$  is further trained by using our adaptive target  $\tilde{y}_i$  generated from the ATG as follows:

$$\sum_i \ell(\tilde{y}_i, f(x_i)). \quad (7)$$

For the training, our new target is generated adaptively with respect to current output at every iteration on-the-fly, and we do not have to prepare additional training data or go through pre-processing. Note that using ATG takes additional 15ms for every iteration on NVIDIA TITAN Xp GPU. Since the image is divided into pieces and then the transformations for each piece are obtained, it is locally more accurate than the single transformation for the entire image. In addition, this strategy can cover spatially varying blur kernels, which may be more adequate for real image SR scenarios.

Some examples of our adaptive target are shown in Fig. 4. We show network outputs  $f(x_i)$  and corresponding adaptive targets  $\tilde{y}_i$  from the pretrained SR model (2-4 columns) and fully trained SR model using our method (5-7 columns). The outputs of the pretrained model are still blurry because the model had difficulty in coping with inconsistent mappings between inputs and outputs caused by diverse blur kernels. Before training with  $\tilde{y}_i$ , generated  $\tilde{y}_i$



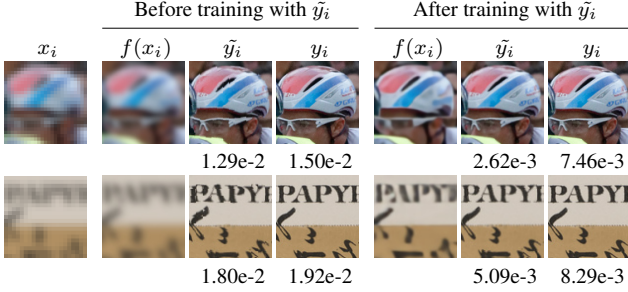


Figure 4. Some examples of our adaptive target  $\tilde{y}_i$  created from original target  $y_i$ . First column shows inputs  $x_i$ , 2-4 columns show the outputs  $f(x_i)$  of a pretrained SR model and their adaptive targets  $\tilde{y}_i$ , and 5-7 columns show those of final SR model trained by our method. We also show MSE of each output compared to the adaptive and the original target. The adaptive targets are not significantly different from the original targets, and penalize the SR network less.

shows some artifacts at patch boundary but they disappear as training progresses. Because our  $\tilde{y}_i$  is created from the near neighborhood, the content does not significantly deviate from the original target  $y_i$ . After the training, the sharpness of the SR results is significantly increased. We also show the MSE of each target and the output, and our targets show lower error as they are designed to less penalize for the acceptable areas. Therefore, training the SR network using  $\tilde{y}_i$  avoids the backpropagation of unnecessary errors that pushes the model to generate exactly the same details as  $y_i$ , which is extremely difficult to achieve. At the same time, our results may have a slightly worse PSNR value because our method allows slightly different outputs compared to  $y_i$ .

## 4. Experiments

We evaluate the effectiveness of our method on two experimental setups: blind SR with diverse blur kernels and non-blind SR with bicubic downsampling for upscale factor 4. For all experiments, we set our SR network  $f$  as pretrained RRDB [39], but any SR networks can be trained with our adaptive target. For fair comparisons with other methods [30, 39, 37, 46, 44, 8, 47, 21], we use DIV2K dataset [1] for training. The dataset contains 800 RGB training images with 2K resolution and covers a large diversity of contents ranging from man-made environments to natural scenes for example-based image SR.

Three metrics are used to quantify the quality of results. The first two are PSNR and structural similarity index (SSIM) [40], which are traditionally used for image quality assessment. They are computed on the pixel domain and are known to not correlate perfectly with the human visual perception [45]. Therefore, we also use the learned perceptual image patch similarity (LPIPS) [45], which is designed to

measure the quality of images from the perspective of human visual perception. Note that lower values are better for LPIPS, and higher values are better for PSNR and SSIM.

For training, we crop the ground truth images into patches of size  $140 \times 140$ , which is divisible by  $p$ , and the size of the corresponding LR input is  $35 \times 35$ . Based on the pretrained SR network  $f$  and ATG, we first train only  $f$  for  $10^5$  iterations using Adam optimizer [16] with learning rate of  $10^{-4}$ . Then, we finetune the whole networks for another  $2 \times 10^4$  iterations with learning rate of  $10^{-5}$ .

In the actual implementation, we empirically found that the original details of  $y_i$  are slightly harmed in the bilinear sampling process for generating  $\tilde{y}_i$ . Therefore, we transform  $f(x_i)$  instead of  $y_i$  and it can be performed simply using the inverse affine matrix  $\theta_i^{-1}$ . The loss Eq. (7) is changed to  $\ell(y_i, \tilde{f}(x_i))$  and we achieve better results as it fully exploits the original details. All experiments are performed in this way.<sup>1</sup>

### 4.1. Diverse Blur Kernels

**Isotropic Gaussian Kernels** First, we conduct blind SR experiments using isotropic Gaussian blur kernels and follow the experimental settings in [8]. The kernel width range is set to  $[0.2, 4.0]$ , and we uniformly sample the kernel width in the above ranges with the step size 0.2 for synthesizing the training image pairs. The GT images are first blurred by the selected blur kernels and then downsampled by bicubic interpolation. For testing, we use Gaussian8 testset proposed in [8]. The images in 5 common SR testsets Set5, Set14, BSDS100 [25], Urban100 [12] and Manga109 [26] are blurred by 8 selected isotropic Gaussian kernels width of  $[1.8, 3.2]$  with the step size 0.2. For example, there is a total of 40 test images for the Set5 testset (5 images  $\times$  8 kernels). Our results are compared with 2 state-of-the-art DNN based blind SR methods IKC [8] and KG [2]. In addition, we also compare with 3 state-of-the-art DNN based non-blind SR methods SRMD [44], MZSR [36], and USRNet [43]. Here, we reproduce the results of the non-blind SR methods using GT kernel information, to assume there is no performance drop due to inaccurate kernel prediction, providing upper bounds.<sup>2</sup> IKC and SRMD are trained only for isotropic Gaussian kernels, while the others can handle anisotropic Gaussian kernels. Specifically, USRNet can also handle motion kernels and KG is designed to deal with any blur kernel.

Quantitative results are shown in Table 2 and the values are averaged for all 8 kernels. Note that we crop every border by 4 pixels to avoid artifacts occurred on borders for all results in the paper. KG first estimates the blur kernel for

<sup>1</sup>Our code is at <https://github.com/yhjo09/AdaTarget>.

<sup>2</sup>Please note that SRMD, MZSR, and USRNet hold an advantage as they use GT kernel information. IKC and KG compute the kernels in their algorithms, and our method does not require any kernel information.

| Method             | Set5         |               |               | Set14        |               |               | BSDS100      |               |               | Urban100     |               |               | Manga109     |               |               |
|--------------------|--------------|---------------|---------------|--------------|---------------|---------------|--------------|---------------|---------------|--------------|---------------|---------------|--------------|---------------|---------------|
|                    | PSNR         | SSIM          | LPIPS         | PSNR         | SSIM          | LPIPS         | PSNR         | SSIM          | LPIPS         | PSNR         | SSIM          | LPIPS         | PSNR         | SSIM          | LPIPS         |
| Bicubic            | 25.85        | 0.7289        | 0.4460        | 24.20        | 0.6222        | 0.5468        | 24.59        | 0.5940        | 0.6445        | 21.68        | 0.5745        | 0.5904        | 22.79        | 0.7097        | 0.4427        |
| KG [2] + ZSSR [34] | -            | -             | -             | -            | -             | -             | -            | -             | -             | 22.06        | 0.6223        | 0.3788        | 24.75        | 0.7786        | 0.2454        |
| IKC [8]            | <b>31.62</b> | <u>0.8808</u> | <u>0.2016</u> | <b>28.18</b> | <u>0.7608</u> | <u>0.3207</u> | <u>27.33</u> | <u>0.7161</u> | <u>0.4174</u> | <u>25.31</u> | <u>0.7487</u> | <u>0.2895</u> | <u>28.82</u> | <u>0.8756</u> | <u>0.1664</u> |
| Ours               | <u>31.58</u> | <b>0.8814</b> | <b>0.1932</b> | <u>28.14</u> | <b>0.7626</b> | <b>0.3122</b> | <b>27.43</b> | <b>0.7216</b> | <b>0.4030</b> | <b>25.72</b> | <b>0.7683</b> | <b>0.2518</b> | <b>29.97</b> | <b>0.8955</b> | <b>0.1286</b> |
| SRMD [44]          | 31.47        | 0.8799        | 0.1898        | 28.11        | 0.7628        | 0.3076        | 27.32        | 0.7200        | 0.4031        | 25.34        | 0.7548        | 0.2682        | 29.80        | 0.8925        | 0.1288        |
| MZSR [36]          | 25.01        | 0.7192        | 0.3851        | 23.60        | 0.6140        | 0.4839        | 24.13        | 0.5866        | 0.5787        | 21.25        | 0.5674        | 0.5071        | 22.08        | 0.7031        | 0.3565        |
| USRNet [43]        | 32.37        | 0.8932        | 0.1786        | 28.68        | 0.7799        | 0.2904        | 27.66        | 0.7349        | 0.3812        | 26.29        | 0.7899        | 0.2287        | 31.02        | 0.9119        | 0.1094        |

Table 2. Quantitative results on Gaussian8 testset for isotropic Gaussian kernels. The top 4 and the bottom 3 are blind and non-blind SR methods respectively. Note that it is an unfair comparison with the non-blind methods as they use GT kernel information. Best values are shown in bold and second best values are underlined. We crop every border by 4px to exclude border artifacts for all results in the paper. Our method shows better or very comparable performance in all testsets.

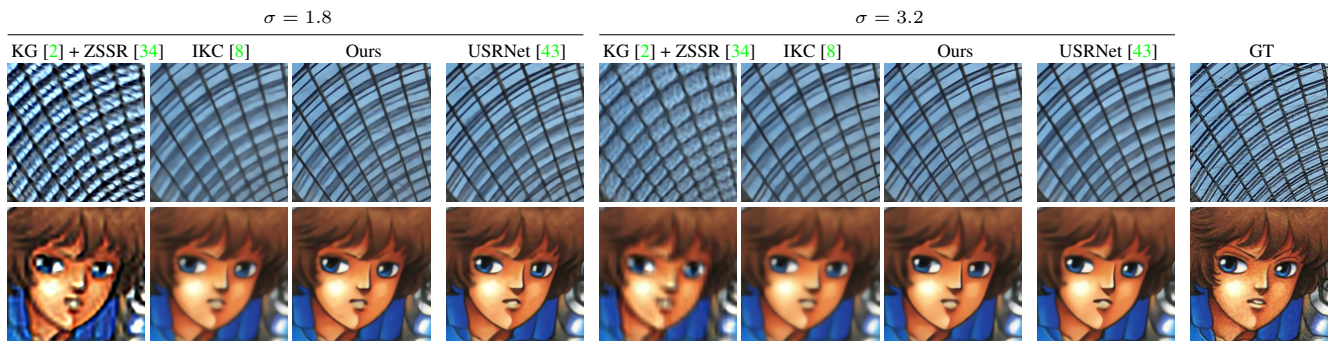


Figure 5. Qualitative results on Gaussian8 testset for isotropic Gaussian kernels. The first 4 columns show the results for the blur kernel width 1.8, the next 4 columns are for the width 3.2, and the last column shows GT image. Our results better restore edges and textures by allowing a variety of acceptable output compared to blind SR methods.

the given image, and the SR part is executed using a non-blind SR method ZSSR [34]. IKC corrects the blur kernel and the result for 7 iterations and selects the best one among them. While both methods feedforward test images several times to output final results, our method works in a single-shot manner with faster runtime. KG + ZSSR, IKC, and our method take 458s, 2.72s, and 0.44s respectively on NVIDIA TITAN Xp GPU for generating a  $1920 \times 1080$  output image. Note that the number of parameters is 0.4M, 9M, and 16.7M respectively. On top of the advantage of the speed, our method shows mostly better PSNR and SSIM values while having much better perceptual quality (LPIPS) compared to IKC and KG + ZSSR.

Producing images with good perceptual quality has become important recently, as evidenced by a flow of GAN-based methods, and our method is able to achieve good perceptual quality while maintaining the original contents. Since our method allows for slightly different outputs, the PSNR and SSIM values computed with the given ground truth target are expected to drop. However, the drop-off in these measures is surprisingly small and our method outperforms others by a good margin in the perceptual quality. Fig. 5 qualitatively shows that our method generates sharper

and visually pleasing images compared to other methods.

**Random Kernels** We also conduct blind SR experiments on random blur kernels by following the experimental settings in [2]. For synthesizing the training images, random kernels are derived from anisotropic Gaussian kernels with uniformly selected kernel width in range  $[0.6, 5]$  for each axis independently, which are rotated by a random angle in  $[-\pi, \pi]$ . In addition, a small amount of uniform multiplicative noise is applied. For the testing, we use DIV2K dataset proposed in [2]. DIV2K has 100 images that are blurred and subsampled with a randomly generated kernel. Each image is from the original 100 validation images of DIV2K dataset [1].

Quantitative results are shown in Table 3. Although IKC is only trained on isotropic Gaussian kernels, it performs better than KG + ZSSR due to the effective model design with a larger capacity. The effectiveness of our method is even more apparent for the random degradation. For the random kernels, we outperform other blind methods in all measures without putting efforts on the estimation and the utilization of kernels. Fig. 6 shows qualitative comparisons for the random kernel experiments. Our method generates impressive results regardless of kernel width sizes and

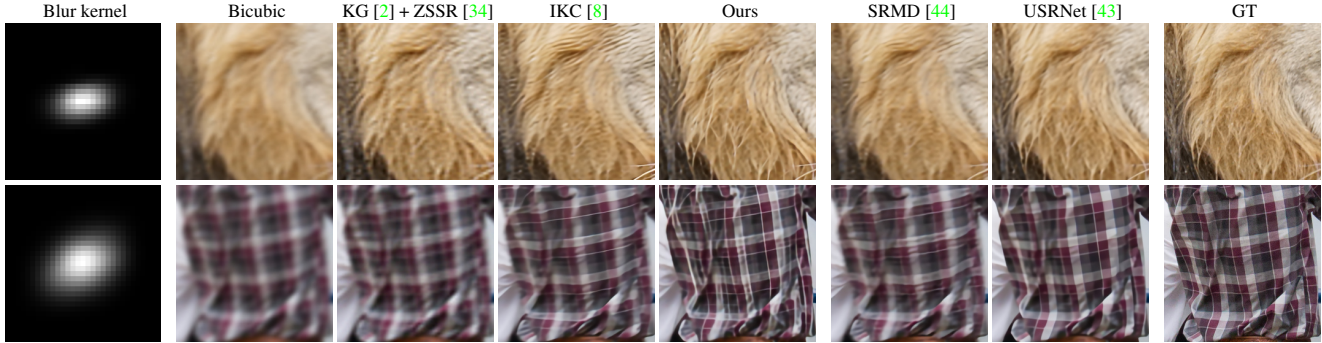


Figure 6. Qualitative results on DIV2KRRK testset for random kernels. Each test image is simulated by different blur kernels shown on the left. Our results show obvious sharpness and clarity compared to other blind SR results regardless of the kernel shapes. Also, our results look more realistic in some examples compared to non-blind SR results.

| Method             | DIV2KRRK     |               |               |
|--------------------|--------------|---------------|---------------|
|                    | PSNR         | SSIM          | LPIPS         |
| Bicubic            | 25.37        | 0.6812        | 0.5635        |
| KG [2] + ZSSR [34] | 26.80        | 0.7319        | 0.4389        |
| IKC [8]            | 27.69        | 0.7658        | 0.3856        |
| Ours               | <b>28.42</b> | <b>0.7854</b> | <b>0.3394</b> |
| SRMD [44]          | 27.63        | 0.7623        | 0.3816        |
| MZSR [36]          | 24.99        | 0.6919        | 0.4096        |
| USRNet [43]        | 29.80        | 0.8148        | 0.3088        |
| Ours (ZSSR)        | 27.47        | 0.7561        | 0.3772        |
| Ours (IKC)         | 28.33        | 0.7799        | 0.3558        |

Table 3. Quantitative results on DIV2KRRK testset for random blur kernels. Our method outperforms others in all metrics except USRNet. The bottom 2 results show that our method also works well with other SR backbone models.

shapes, restoring sharp edges clearly and without blurriness.

## 4.2. Bicubic Downsampling

In this section, training images are simulated by bicubic downsampling and we train two models – one with and the other without GAN. For the GAN training, we use an adversarial loss [42], the perceptual loss [14], and our adaptive target loss altogether. For the perceptual loss, our adaptive target is used for computing the feature response of the VGG network [35]. For the test, we use 5 common SR testsets Set5, Set14, BSDS100 [25], Urban100 [12] and Manga109 [26].

In Table 4, the first 2 rows show the comparisons of methods without GAN and the other rows show methods with GAN. When GANs are not used, our approach shows a minor effect with a very small drop in PSNR values and slightly improved SSIM and LPIPS values. This is expected as the measures are computed only using the given ground truth but our method allows slightly different outputs, and

also as there are less ambiguities in this fixed blur kernel scenario compared to the blind case.

However, we observe that our method shows meaningful improvements when combined with GAN for realistic and perceptual SR. The adversarial loss is an unsupervised loss that creates arbitrary details for realistic outputs. However, the arbitrary details that the adversarial loss generates can conflict with real details from the original training target when computing the supervised losses (the perceptual loss and our adaptive loss). Our adaptive target reduces the conflict as it is created to less penalize the current output by design. As a result, our method achieves the better or very comparable LPIPS values for all testsets, and this also shows up with good visual results in Fig. 7. Compared to our backbone model, ESRGAN, our method restores the content much better (more clear text), as the network is less susceptible to errors from off-target details. Our method also achieves better PSNR and SSIM values by a large margin at the same time, showing the ability to maintain the original contents better than other GAN-based methods.

## 4.3. Analysis

**Localization Network** Here, we conduct experiments to analyze the effect of the learned parameters of the localization network. We incrementally add a basic transformation when pretraining the localization network – Tr (translation), Tr + Rot (rotation), Tr + Rot + Shear, and Tr + Rot + Shear + Scale. We apply small amounts of random shearing and random scaling within range  $[-10, 10]$  degree and  $[0.9, 1.1]$  respectively. We also experiment with an enlarged search space  $s = 11$  using our default transformation setting (Tr + Rot). We additionally implement nearest neighbor (NN) search as a baseline approach for generating the adaptive target instead of using the localization network (Fig. 3). For each  $p \times p$  piece, the NN approach finds the minimal error area on the corresponding  $s \times s$  piece in integer pixel step.

The overall experimental process is the same as the ran-



| Method          | Set5         |               |               | Set14        |               |               | BSDS100      |               |               | Urban100     |               |               | Manga109     |               |               |
|-----------------|--------------|---------------|---------------|--------------|---------------|---------------|--------------|---------------|---------------|--------------|---------------|---------------|--------------|---------------|---------------|
|                 | PSNR         | SSIM          | LPIPS         | PSNR         | SSIM          | LPIPS         | PSNR         | SSIM          | LPIPS         | PSNR         | SSIM          | LPIPS         | PSNR         | SSIM          | LPIPS         |
| RRDB [39]       | <b>32.68</b> | 0.8999        | 0.1690        | <b>28.88</b> | 0.7891        | 0.2717        | <b>27.82</b> | 0.7444        | 0.3586        | <b>27.02</b> | 0.8146        | 0.1955        | <b>31.57</b> | <b>0.9185</b> | 0.0976        |
| Ours (RRDB)     | 32.62        | <b>0.9004</b> | <b>0.1655</b> | 28.80        | <b>0.7913</b> | <b>0.2672</b> | 27.80        | <b>0.7486</b> | <b>0.3505</b> | 26.97        | <b>0.8161</b> | <b>0.1899</b> | 31.42        | 0.9181        | <b>0.0954</b> |
| SRGAN [18]      | 29.41        | 0.8345        | 0.0817        | 26.01        | 0.6932        | 0.1471        | 25.18        | 0.6401        | 0.1846        | -            | -             | -             | -            | -             | -             |
| EnhanceNet [33] | 28.56        | 0.8093        | 0.0997        | 25.67        | 0.6759        | 0.1601        | 24.93        | 0.6259        | 0.2035        | 23.54        | 0.6926        | 0.1638        | 20.44        | 0.6703        | 0.1270        |
| SRFeat [30]     | 30.05        | 0.8465        | 0.0685        | 26.60        | 0.7106        | <b>0.1299</b> | 25.08        | 0.6574        | 0.1753        | 24.52        | 0.7306        | 0.1346        | 28.15        | 0.8525        | 0.0670        |
| NatSR [37]      | 30.99        | 0.8610        | 0.0931        | <b>27.42</b> | 0.7330        | 0.1737        | 26.44        | 0.6831        | 0.2122        | 25.46        | 0.7605        | 0.1505        | 29.50        | 0.8812        | 0.0732        |
| RankSRGAN [46]  | 29.66        | 0.8364        | 0.0703        | 26.36        | 0.7003        | 0.1356        | 25.45        | 0.6484        | 0.1760        | 24.47        | 0.7288        | 0.1383        | 27.90        | 0.8488        | 0.0760        |
| SPSR [24]       | 30.40        | 0.8427        | <b>0.0613</b> | 26.53        | 0.7101        | 0.1308        | 25.51        | 0.6576        | 0.1628        | 24.80        | 0.7466        | 0.1188        | 28.61        | 0.8579        | 0.0671        |
| ESRGAN [39]     | 30.44        | 0.8506        | 0.0736        | 26.22        | 0.6967        | 0.1324        | 25.42        | 0.6543        | 0.1643        | 24.36        | 0.7333        | 0.1236        | 28.41        | 0.8591        | 0.0645        |
| Ours (ESRGAN)   | <b>31.55</b> | <b>0.8708</b> | 0.0644        | 27.31        | <b>0.7389</b> | <b>0.1299</b> | <b>26.45</b> | <b>0.6914</b> | <b>0.1611</b> | <b>25.78</b> | <b>0.7762</b> | <b>0.1105</b> | <b>29.90</b> | <b>0.8828</b> | <b>0.0549</b> |

Table 4. Quantitative results on 5 common SR testsets for bicubic downsampling. The top 2 rows are the methods without GAN, and the rest are the methods with GAN. Our method achieves better or very comparable scores in all metrics when combined with GAN for realistic SR.



Figure 7. Qualitative results for perceptual SR. Our method generates consistent details for text which GAN-based methods are difficult to restore. Please see supplementary for more results.

| Learned transformations           | DIV2KRK      |               |               |
|-----------------------------------|--------------|---------------|---------------|
|                                   | PSNR         | SSIM          | LPIPS         |
| Integer step Tr (NN search)       | 27.75        | 0.7772        | 0.3195        |
| Tr                                | <b>28.42</b> | 0.7832        | 0.3469        |
| Tr + Rot (Ours)                   | <b>28.42</b> | 0.7854        | 0.3394        |
| Tr + Rot ( $p = 7$ and $s = 11$ ) | 28.27        | 0.7778        | 0.3679        |
| Tr + Rot + Shear                  | 28.39        | 0.7851        | 0.3402        |
| Tr + Rot + Shear + Scale          | 28.26        | <b>0.7855</b> | <b>0.3382</b> |

Table 5. Effect of differently learned localization networks.

dom kernel experiment. We compute PSNR, SSIM, and LPIPS for the DIV2KRK testset in Table 5. The NN search approach achieves a better LPIPS value but worse PSNR value compared to the STN based methods. We infer that the worse PSNR value comes from the matching which is not performed in sub-pixel accuracy, while the better LPIPS value comes from no detail loss induced by sub-pixel sampling. For the STN based methods, the performance gap is not very significant. We choose the Tr + Rot model as our default setting, because it shows the better PSNR with very comparable LPIPS. If the search space becomes large, it causes difficulty in restoring original contents and shows degraded quality as an imperfect output is more likely to match with areas outside the original target location.

**Generalization** We validate our method with other SR backbone networks to test the generalization ability – ones

from IKC [8] and ZSSR [34]. The overall experimental process is the same as the random kernels experiment. The quantitative results are shown in the bottom of Table 3, and we verify that our method increases the performance in all metrics by a good margin compared to the original models.

## 5. Conclusion

We proposed a novel way to deal with the ill-posed nature of SR by introducing the adaptive target. Our method can be applied to existing training data and network architectures without any modification, and without adding too much training time. We verified that our results outperform previous state-of-the-art methods, especially in generating perceptually pleasing images. For the blind SR, our method holds another advantage that no additional effort is required to find accurate blur kernels. In the future, we would like to extend our work to consider multi-frame for computing a new target. Additionally, we plan to extend our work for deblurring as it inherently possesses the ill-posedness as well.

**Acknowledgement** This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2014-3-00123, Development of High Performance Visual BigData Discovery Platform for Large-Scale Realtime Data Analysis, and No. 2020-0-01361, Artificial Intelligence Graduate School Program (Yonsei University)).



## References

- [1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPR Workshops*, July 2017. 5, 6
- [2] Sefi Bell-Kligler, Assaf Shocher, and Michal Irani. Blind super-resolution kernel estimation using an internal-gan. In *NeurIPS*, pages 284–293, 2019. 2, 5, 6, 7
- [3] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, pages 184–199. Springer, 2014. 2
- [4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. volume 70 of *PMLR*, pages 1126–1135, 2017. 2
- [5] William T Freeman, Thouis R Jones, and Egon C Pasztor. Example-based super-resolution. *IEEE Computer graphics and Applications*, 22(2):56–65, 2002. 1
- [6] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *ICCV*, pages 349–356. IEEE, 2009. 1
- [7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014. 2
- [8] Jinjin Gu, Hannan Lu, Wangmeng Zuo, and Chao Dong. Blind super-resolution with iterative kernel correction. In *CVPR*, pages 1604–1613, 2019. 2, 5, 6, 7, 8
- [9] Yong Guo, Jian Chen, Jingdong Wang, Qi Chen, Jiezhong Cao, Zeshuai Deng, Yanwu Xu, and Minghui Tan. Closed-loop matters: Dual regression networks for single image super-resolution. *arXiv preprint arXiv:2003.07018*, 2020. 2
- [10] Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. In *NeurIPS*, pages 1799–1807, 2012. 3
- [11] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution. In *CVPR*, pages 1664–1673, 2018. 2
- [12] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, pages 5197–5206, 2015. 1, 5, 7
- [13] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NeurIPS*, pages 2017–2025, 2015. 4
- [14] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711. Springer, 2016. 7
- [15] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, pages 1646–1654, 2016. 2
- [16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5
- [17] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, pages 624–632, 2017. 2
- [18] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, July 2017. 2, 8
- [19] Stefan Lee, Senthil Purushwalkam Shiva Prakash, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. Stochastic multiple choice learning for training diverse deep ensembles. In *NeurIPS*, pages 2119–2127, 2016. 3
- [20] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Interactive image segmentation with latent diversity. In *CVPR*, pages 577–585, 2018. 3
- [21] Zhen Li, Jinglei Yang, Zheng Liu, Xiaomin Yang, Gwanggil Jeon, and Wei Wu. Feedback network for image super-resolution. In *CVPR*, pages 3867–3876, 2019. 2, 5
- [22] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, pages 136–144, 2017. 2
- [23] Andreas Lugmayr, Martin Danelljan, Luc Van Gool, and Radu Timofte. Srfrow: Learning the super-resolution space with normalizing flow. In *ECCV*, 2020. 2
- [24] Cheng Ma, Yongming Rao, Yean Cheng, Ce Chen, Jiwen Lu, and Jie Zhou. Structure-preserving super resolution with gradient guidance. In *CVPR*, 2020. 2, 8
- [25] David Martin, Charles Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, volume 2, pages 416–423. IEEE, 2001. 5, 7
- [26] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multimedia Tools and Applications*, 76(20):21811–21838, 2017. 5, 7
- [27] Yiqun Mei, Yuchen Fan, Yuqian Zhou, Lichao Huang, Thomas S Huang, and Honghui Shi. Image super-resolution with cross-scale non-local attention and exhaustive self-exemplars mining. In *CVPR*, pages 5690–5699, 2020. 2
- [28] Ben Niu, Weilei Wen, Wenqi Ren, Xiangde Zhang, Lianping Yang, Shuzhen Wang, Kaihao Zhang, Xiaochun Cao, and Haifeng Shen. Single image super-resolution via a holistic attention network. In *CVPR*, 2020. 2
- [29] Seobin Park, Jinsu Yoo, Donghyeon Cho, Jiwon Kim, and Tae Hyun Kim. Fast adaptation to super-resolution networks via meta-learning. In *ECCV*, 2020. 2
- [30] Seong-Jin Park, Hyeongseok Son, Sunghyun Cho, Ki-Sang Hong, and Seungyong Lee. Srfeat: Single image super-resolution with feature discrimination. In *ECCV*, pages 439–455, 2018. 2, 5, 8
- [31] Yajun Qiu, Ruxin Wang, Dapeng Tao, and Jun Cheng. Embedded block residual network: A recursive restoration model for single-image super-resolution. In *ICCV*, pages 4180–4189, 2019. 2
- [32] Mohammad Saeed Rad, Behzad Bozorgtabar, Urs-Viktor Marti, Max Basler, Hazim Kemal Ekenel, and Jean-Philippe Thiran. Srobb: Targeted perceptual loss for single image super-resolution. In *ICCV*, October 2019. 2

- [33] Mehdi S. M. Sajjadi, Bernhard Scholkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *ICCV*, Oct 2017. 2, 8
- [34] Assaf Shocher, Nadav Cohen, and Michal Irani. “zero-shot” super-resolution using deep internal learning. In *CVPR*, pages 3118–3126, 2018. 2, 6, 7, 8
- [35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 7
- [36] Jae Woong Soh, Sunwoo Cho, and Nam Ik Cho. Meta-transfer learning for zero-shot super-resolution. In *CVPR*, pages 3516–3525, 2020. 2, 5, 6, 7
- [37] Jae Woong Soh, Gu Yong Park, Junho Jo, and Nam Ik Cho. Natural and realistic single image super-resolution with explicit natural manifold discrimination. In *CVPR*, June 2019. 2, 5, 8
- [38] Radu Timofte, Vincent De Smet, and Luc Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV*, pages 111–126. Springer, 2014. 1
- [39] 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 *ECCV Workshops*, September 2018. 2, 5, 8
- [40] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004. 5
- [41] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE TIP*, 19(11):2861–2873, 2010. 1
- [42] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018. 7
- [43] Kai Zhang, Luc Van Gool, and Radu Timofte. Deep unfolding network for image super-resolution. In *CVPR*, pages 3217–3226, 2020. 5, 6, 7
- [44] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Learning a single convolutional super-resolution network for multiple degradations. In *CVPR*, pages 3262–3271, 2018. 2, 5, 6, 7
- [45] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5
- [46] Wenlong Zhang, Yihao Liu, Chao Dong, and Yu Qiao. Ranksrgan: Generative adversarial networks with ranker for image super-resolution. In *ICCV*, October 2019. 2, 5, 8
- [47] 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*, pages 286–301, 2018. 2, 5
- [48] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, pages 2472–2481, 2018. 2
- [49] Ruofan Zhou and Sabine Susstrunk. Kernel modeling super-resolution on real low-resolution images. In *ICCV*, pages 2433–2443, 2019. 2