

# Toward Real-World Super-Resolution via Adaptive Downsampling Models

Sanghyun Son\*, Jaeha Kim\*, Wei-Sheng Lai, Ming-Hsuan Yang, and Kyoung Mu Lee *Fellow, IEEE*,

**Abstract**—Most image super-resolution (SR) methods are developed on synthetic low-resolution (LR) and high-resolution (HR) image pairs that are constructed by a predetermined operation, e.g., bicubic downsampling. As existing methods typically learn an inverse mapping of the specific function, they produce blurry results when applied to real-world images whose exact formulation is different and unknown. Therefore, several methods attempt to synthesize much more diverse LR samples or learn a realistic downsampling model. However, due to restrictive assumptions on the downsampling process, they are still biased and less generalizable. This study proposes a novel method to simulate an unknown downsampling process without imposing restrictive prior knowledge. We propose a generalizable low-frequency loss (LFL) in the adversarial training framework to imitate the distribution of target LR images without using any paired examples. Furthermore, we design an adaptive data loss (ADL) for the downsampler, which can be adaptively learned and updated from the data during the training loops. Extensive experiments validate that our downsampling model can facilitate existing SR methods to perform more accurate reconstructions on various synthetic and real-world examples than the conventional approaches.

**Index Terms**—Image super-resolution, image downsampling, unsupervised learning

## 1 INTRODUCTION

IMAGE super-resolution (SR), which aims to reconstruct a high-resolution (HR) image from a low-resolution (LR) input, plays an essential role in computer vision and digital photography. There exist numerous applications, including enhancing the details and photorealism of an image [1], high-quality editing [2], and breaking the sensor limitation of mobile cameras [3]. Recently, a plethora of SR methods have been developed on the basis of deep CNNs [1], [4], [5], [6] and large-scale datasets [6], [7]. However, state-of-the-art methods [8], [9], [10], [11] do not generalize well to the real-world inputs even they perform relatively well on synthesized, e.g., bicubic-downsampled, LR images. In overcoming this issue, few recent approaches [12], [13], [14], [15] have collected high-quality pairs of real-world LR and HR examples to learn their SR models. Nevertheless, such an acquisition process remains to be challenging due to outdoor scene dynamics and spatial misalignments [12].

Conventional SR methods synthesize various LR samples  $\mathbf{I}_{LR}$  from ground-truth HR images  $\mathbf{I}_{HR}$  by the following:

$$\mathbf{I}_{LR} = (\mathbf{I}_{HR} * k)_{\downarrow s} + n, \quad (1)$$

where  $k \in \mathbb{R}^2$  is a 2D degradation kernel,  $*$  is a spatial convolution,  $\downarrow_s$  is a decimation with a stride  $s$ , and  $n$  is a noise term. The decimation operator corresponds to direct downsampling mentioned in the super-resolution literature [16]. With a specific assumption of blur kernels, e.g., variants of Gaussian [16], [17], [18], LR and HR pairs can be synthesized to train the following SR models. However,

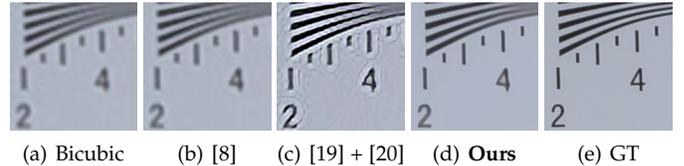


Fig. 1.  $\times 4$  SR results on a real-world LR image. (a) LR image magnified with bicubic interpolation. (b) Result of RRDB [8]. (c) Result of KernelGAN [19] + ZSSR [20]. (d) Our unsupervised approach (ADL + RRDB) reconstructs a sharp and visually pleasing output without artifacts and aliasing compared with the existing methods. (e) Ground-truth patch from RealSR-V3 [14]. Images are cropped from 'Canon/045.png'.

such prior typically limit the kernel space, and the synthesized LR images may not reflect the distribution of real-world inputs [13]. Therefore, the learned SR models become less generalizable toward arbitrary real-world input images.

On the other hand, recent unsupervised methods simulate real-world LR samples that contain unknown noise [21], [22] and artifacts [23], [24]. Without using a paired dataset, they first learn a downsampling model under adversarial training frameworks [25] to imitate the distribution of real-world images. The following SR models can then be trained in a supervised manner on the simulated dataset to deliver accurate reconstruction results on the real-world inputs. One of the challenges in such methods arises from preserving image content across different scales, i.e., HR and LR, while learning the downsampling model. Existing approaches deal with this problem using a predetermined downsampling operator, e.g., bicubic downsampling, in their objective functions and constrain the simulated LR images not to deviate much from the known formulations. However, the manual selection of the operator can introduce a bias in the unsupervised learning framework, which can also act as a restrictive prior if the ground-truth downsampling model is much different from the used one. While KernelGAN [19] alleviates the issue by estimating a low-dimensional downsampling kernel  $k$  in (1) from an LR image  $\mathbf{I}_{LR}$ , various regularization terms need to be applied to restrict the diversity of the possible kernel space.

- \*Authors contributed equally.
- S. Son, J. Kim, and K. M. Lee are with the Department of ECE & ASRI, Seoul National University, Seoul, Korea, 08826. E-mail: {thstkdgus35, hjkim2, kyoungmu}@snu.ac.kr
- W.-S. Lai is with Google. E-mail: wslai@google.com
- M.-H. Yang is with Google, UC Merced, and Yonsei University. E-mail: minghsuan@google.com

Therefore, we propose an effective way of imitating the real-world LR samples of an unknown distribution to address the aforementioned issues. Similar to the previous unsupervised methods [19], [23], we also train a down-sampling CNN to simulate the LR images in our target distribution. However, rather than formulating the objective function with a handcrafted downsampling operator, we propose a novel and generalizable low-frequency loss (LFL) that does not pose substantial bias. Our LFL facilitates the downsampling model to learn much more diverse and precise functions without being constrained to a specific prior assumption. Furthermore, we develop an adaptive data loss (ADL) that iteratively adjusts the training objective for the given dataset and stabilizes the learning process. As shown in Fig. 1, our unsupervised learning framework is straightforward, effective, and generalizable to arbitrary downsampling models. Extensive experiments validate that the SR models learned on our downsampled images perform favorably on synthetic and real-world LR images. The contributions of this study can be organized threefold:

- We present a novel unsupervised learning framework to learn an unknown downsampling process without using any HR and LR image pairs.
- We propose LFL and ADL to simulate accurate and realistic LR samples from HR images without relying on any predetermined downsampling operators.
- We demonstrate that the proposed method can be easily integrated with existing SR frameworks and achieve much better results on synthetic and real-world images.

## 2 RELATED WORK

### 2.1 SR on bicubic downsampled images

With the success of SRCNN [4], several CNN-based methods have been developed for image SR. As one of the most influential studies, VDSR [26] has proposed a novel residual learning strategy to train a deep network and inspired lots of following methods [1], [6], [27], [28]. Earlier works primarily focus on improving the network designs, such as pixel shuffling [29], progressive upsampling [5], [30], [31], dense connections [11], [32], [33], recursive structures [34], [35], [36], and back-projection [10]. Recent approaches utilize the attention [9], [37], [38], while designing architectures for efficient inference [39], [40], [41] is considered essential as well. From the perspective of image realism, several methods introduce perceptual loss [42], [43], [44], [45] to synthesize photorealistic textures [1], [8], [46], [47]. However, the existing methods are typically trained on synthesized image pairs in which LR inputs are generated using conventional bicubic interpolation from HR targets. While state-of-the-art algorithms perform impressively well when training and test distributions are matched, i.e., test inputs are also downsampled with the same operator, they cannot be fully generalized to arbitrary in-the-wild LR images [20], [48].

### 2.2 Synthesizing diverse LR images for SR

For practical SR application, it is essential to determine how to generate LR images [49] so that a supervised SR model can be trained without real-world LR and HR image pairs. Several approaches have synthesized diverse LR images

with multiple degradations to train their SR algorithms, assuming that the generalization on such examples can improve SR performance on arbitrary inputs. SRMD [16] considers the formation of LR images under various down-sampling kernels  $k$  in (1). It can reconstruct HR images from diverse types of LR inputs, using off-the-shelf methods [19], [50], [51] to predict a candidate kernel  $k$  from a given LR input. USRNet [52] further allows diversity to the down-sampling kernel  $k$  and can deliver clean SR results even when LR inputs are corrupted with motion blur and noise.

Furthermore, recent methods [17], [18], [53] present unified frameworks to jointly estimate the kernel  $k$  and reconstruct visually pleasing results from an arbitrary LR image. However, since considering all possible forms of downsampling operation is not practical, the candidate kernels in such methods are often simplified to variants of 2D Gaussian. Recent studies have demonstrated that such approximations may not hold for actual LR images [13], [54] in the wild, thus making the abovementioned SR algorithms less generalizable. In this study, we demonstrate that existing approaches [17], [18] do not perform well on inputs from unknown downsampling kernels or real-world images, and our method provides better generalization.

### 2.3 Learning to simulate real-world LR images

Instead of synthesizing LR images from some handcrafted formulations, numerous approaches [21], [22], [23], [24] have adopted adversarial training [25] to simulate the unknown distributions of real-world LR images using down-sampler CNNs. These methods have shown impressive performance when dealing with unknown noise [21], [22] and artifacts [23], [24] in real-world LR images. Considering the definition of downsampling, one of the required characteristics of such methods is to preserve the contents of HR input and generate a feasible LR image. Therefore, predetermined downsampling operators [22], [23], [24] and cyclic architecture [55] are used to guide the generated LR images not deviating much from the desired outputs. However, a significant limitation of this formulation is that the necessity of estimating an accurate *downsampling* process is often considered less important. In particular, the handcrafted operators may significantly differ from the unknown downsampling function and bias the following downsampler, making the model less effective in estimating the actual operators rather than noise and artifacts.

On the other hand, KernelGAN [19] is designed to directly predict the degradation kernel  $k$ , which is used to generate the given LR image  $I_{LR}$ . The estimated kernel is then used to synthesize LR and HR pairs for the following SR model [20]. In addressing the ill-posed problem of finding the kernel  $k$  in (1), several optimization constraints are assumed, such as patch recurrences [50] in a single image, deep linear generator, and various prior knowledge on physically meaningful kernels. However, this approach may not handle practical cases in which such strong assumptions do not hold. While Ji *et al.* [56] have extended the approach to a set of LR images, several prior terms for the appropriate degradation kernel still act as a bottleneck for generalization. On the contrary, our LFL and ADL are designed to reduce inherent bias from adopting a specific downsampling operator or strong kernel priors.

## 2.4 Paired datasets for real-world SR

Limitations of existing SR methods arise from difficulties in constructing the real-world dataset. Few approaches capture the paired dataset [13], [14], [15] by precisely manipulating camera parameters in which images captured from long and short focal lengths are labeled as HR and LR samples, respectively. Zhang *et al.* [12] introduces SR-RAW based on raw images and contextual bilateral loss to handle misalignments in the real-world pairs. Xu *et al.* [57] utilizes raw and color images jointly in their model for effective real-world SR. Those image pairs can be used to learn the real-world SR models to some extent. Nevertheless, they still suffer from a lack of scene diversity, misalignments, dynamic motions, and scalability issues. To overcome the several challenges in acquiring realistic SR datasets, we synthesize accurate HR and LR pairs from unpaired examples. While we assume that a set of LR images undergo the same or similar formation process, the data collection is much easier since careful alignment and delicate post-processing are not required.

## 3 LEARNING TO DOWNSAMPLE

In conventional frameworks, mismatches between the handcrafted kernel space and real-world downsampling model [13], [54] makes the following SR networks less generalizable. Thus, we develop an unsupervised learning framework to accurately simulate the LR samples  $\mathbf{I}_{LR} \in \mathcal{I}_{LR}$  from the *unpaired* HR images  $\mathbf{I}_{HR} \in \mathcal{I}_{HR}$ . The following SR model can then be trained to reconstruct the HR results from the given LR dataset  $\mathcal{I}_{LR}$ . For simplicity, we assume that the LR and HR images have spatial resolutions of  $H \times W$  and  $sH \times sW$ , respectively, for a downsampling factor  $s$ .

### 3.1 Learning an unknown downsampling process

We synthesize LR images under the generalized formulation as  $\mathbf{I}_{LR} = \mathcal{D}^*(\mathbf{I}_{HR}^*)$ , where  $\mathbf{I}_{HR}^* \in \mathcal{I}_{HR}^*$  is the latent HR samples from the distribution  $\mathcal{I}_{HR}^*$  and  $\mathcal{D}^*$  is an unknown downsampling operator. The goal is to learn an SR model  $\mathcal{S}$ , which can reconstruct a high-quality HR image from the given LR image  $\mathbf{I}_{LR} \in \mathcal{I}_{LR}$ . However, it is not straightforward to learn the upsampling function directly as the corresponding ground-truth HR images  $\mathcal{I}_{HR}^*$  are unavailable. Thus, we first learn a downsampling model  $\mathcal{D}$  in an unsupervised manner, so that the distribution of the synthesized images  $\mathbf{I}_{Down} = \mathcal{D}(\mathbf{I}_{HR})$  is close to the distribution of the target LR samples  $\mathcal{I}_{LR}$ . By using the generated pairs of  $(\mathbf{I}_{Down}, \mathbf{I}_{HR})$ , our SR model can be trained to reconstruct HR images from the given LR distribution  $\mathcal{I}_{LR}$  in a fully supervised manner.

To learn the downsampling function  $\mathcal{D}$ , we adopt adversarial training framework [25] to jointly optimize the downsampler CNN  $\mathcal{D}(\cdot; \theta_D)$  and the discriminator CNN  $\mathcal{F}(\cdot; \theta_F)$ . Then, we formulate the downsampling and discriminator objectives,  $\mathcal{L}_D$  and  $\mathcal{L}_F$ , as follows:

$$\begin{aligned} \mathcal{L}_D &= \alpha \mathcal{L}_{data} + \mathcal{L}_{adv} = \alpha \mathcal{L}_{data} - \mathbb{E}[\log \mathcal{F}(\mathbf{I}_{Down})], \\ \mathcal{L}_F &= -\mathbb{E}[\log \mathcal{F}(\mathbf{I}_{LR})] + \mathbb{E}[\log(1 - \mathcal{F}(\mathbf{I}_{Down}))], \end{aligned} \quad (2)$$

where  $\mathcal{L}_{data}$  is the data loss,  $\mathcal{L}_{adv}$  is the adversarial loss [25], and  $\alpha$  is a hyperparameter. If the learned downsampling model can accurately synthesize LR images from  $\mathcal{I}_{HR}$ , i.e., the distribution of  $\mathbf{I}_{Down}$  and  $\mathcal{I}_{LR}$  are approximately the

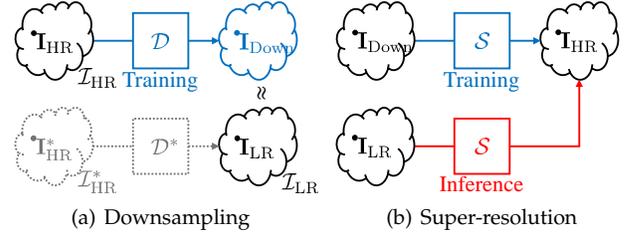


Fig. 2. **Our two-stage approach for unpaired SR.** (a) We first optimize a downsampling model  $\mathcal{D}$  to synthesize  $\mathbf{I}_{LR}$  from  $\mathbf{I}_{HR}$ . The primary goal is to learn the distribution of downsampled images rather than a proper downsampling function. (b) Using generated pairs, we train the SR model  $\mathcal{S}$ , which can also be generalized to the target LR images  $\mathbf{I}_{LR}$ . Dotted lines represent latent components that are not available in the entire learning process. Blue items show learned elements in each stage, and red elements denote the actual goal we want to achieve.

same, the following SR model can be generalized on  $\mathcal{I}_{LR}$  by learning from a set of training pairs  $(\mathbf{I}_{Down}, \mathbf{I}_{HR})$ . Fig. 2 shows the overall pipeline of our method, which learns the downsampling and super-resolution models consecutively.

For simplicity, we assume that the target LR images  $\mathbf{I}_{LR}$  are not corrupted with noise, where the term  $n$  in (1) is ignored. The primary reason is that the noise can be a discriminative feature between the real LR and downsampled images in adversarial training. Since we do not include randomness in our downsampler architecture, such behavior also prevents the proposed method from learning an accurate downsampling function. In Section 4.7, we discuss the effect of real-world noise in the proposed framework.

### 3.2 Data constraint in the downsampling model

In practice, the actual formulation of the given LR images, i.e., the ground-truth downsampling model, is unknown. Thus, we introduce the adversarial loss  $\mathcal{L}_{adv}$  to enforce the downsampled images  $\mathbf{I}_{Down}$  to follow a target distribution  $\mathcal{I}_{LR}$  without using ground-truth LR images. However, unlike the other image generation tasks [58], appropriate constraints are also required to generate faithful LR samples to the given HR counterparts and preserve input contents. In particular, low-level information of a given image, e.g., pixel colors and edge structures, should not be changed during the downsampling, as shown in Fig. 3(a) and (b). Thus, the appropriate formulation of the data term  $\mathcal{L}_{data}$  in (2) plays a critical role in preserving the image content across different scales. A widely-used approach is to define the data loss  $\mathcal{L}_{data}$  with a known operator  $\mathcal{R}_{HR}$ , such as bicubic downsampling or  $s \times s$  average pooling [24], as follows:

$$\begin{aligned} \mathcal{L}_{data} &= \|\mathcal{R}_{HR}(\mathbf{I}_{HR}) - \mathcal{D}(\mathbf{I}_{HR})\|_1, \\ &= \|\mathcal{R}_{HR}(\mathbf{I}_{HR}) - \mathbf{I}_{Down}\|_1. \end{aligned} \quad (3)$$

That is, a reference example  $\mathcal{R}_{HR}(\mathbf{I}_{HR})$  constrains the generated LR sample  $\mathcal{D}(\mathbf{I}_{HR})$  to be a feasible downsampled image. A recent method from Maeda [22] has also combined the bicubic downsampling operator  $\mathcal{B}$  and image-to-image translation CNN  $\mathcal{G}$  in their downsampling model so that  $\mathcal{D} = \mathcal{G} \circ \mathcal{B}$ . Under the such configuration, the translator network  $\mathcal{G}$  is trained to maintain the consistency between its input and output which corresponds to  $\mathcal{R}_{HR} = \mathcal{B}$  in (3).

In (3), the data term  $\mathcal{L}_{data}$  enforces the downsampled images  $\mathbf{I}_{Down}$  to be close to references  $\mathcal{R}_{HR}(\mathbf{I}_{HR})$ . Such a formulation contributes to preserve the image content and

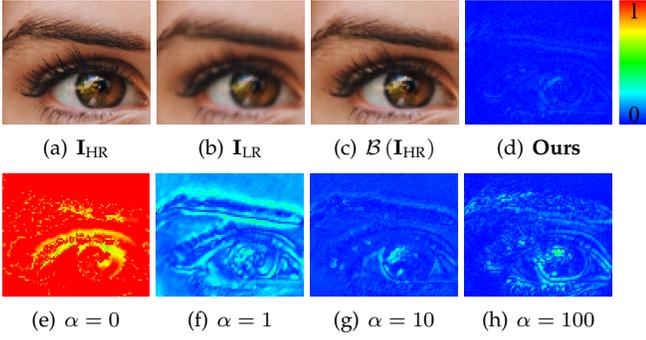


Fig. 3. **Differences between the ground-truth and learned LR images under various configurations.** (a) A reference HR. (b) A ground-truth LR patch  $\mathbf{I}_{LR} = (\mathbf{I}_{HR} * k)_{\downarrow 2}$  we want to synthesize, where the kernel  $k$  is unknown. (c) The corresponding bicubic-downsampled LR which is *different* from  $\mathbf{I}_{LR}$ . (d) The absolute difference between  $\mathbf{I}_{LR}$  and generated LR from our downsampling model is visualized with color-coding, where red pixels indicate large differences. (e)–(h) Difference between  $\mathbf{I}_{LR}$  and outputs from the learned downsampler under (2), where the bicubic downsampling operator is used for  $\mathcal{L}_{data}$ . Difference maps are normalized for better visualization. See more details in Section 4.2.

facilitate the training process for generating LR images. Nevertheless, optimizing the data term  $\mathcal{L}_{data}$  in (2) may bias the learned model toward the used operator  $\mathcal{R}_{HR}$ . The bias may conflict with the adversarial training objective  $\mathcal{L}_{adv}$  if the distribution of the downsampled images  $\mathcal{R}_{HR}(\mathbf{I}_{HR})$  deviate significantly from the target distribution  $\mathcal{I}_{LR}$ .

Fig. 3 illustrates an example to demonstrate the negative effect of using a predetermined downsampling operator, e.g., bicubic kernel  $\mathcal{B}$ , in the data term  $\mathcal{L}_{data}$ . The target LR images  $\mathbf{I}_{LR}$  are generated using a different kernel  $k$ , where  $\mathcal{B}(\mathbf{I}_{HR}) \neq (\mathbf{I}_{HR} * k)_{\downarrow s}$  for an arbitrary HR image  $\mathbf{I}_{HR}$  shown in Fig. 3(a)–(c). Then, we jointly minimize the data and adversarial loss terms in (2) under different  $\alpha$  values so that the downsampling model can be close to the target distribution  $\mathcal{I}_{LR}$ . Fig. 3(e)–(h) illustrate differences between the actual LR and downsampled image  $|\mathbf{I}_{LR} - \mathbf{I}_{Down}|$  with a varying  $\alpha$ . If the data term  $\mathcal{L}_{data}$  is not used, i.e.,  $\alpha = 0$ , the adversarial loss  $\mathcal{L}_{adv}$  is solely optimized in the training so that  $\mathcal{D}(\mathbf{I}_{HR}) \in \mathcal{I}_{LR}$ . As shown in Fig. 3(e), the downsampled image does not preserve the original colors and becomes inconsistent with the input  $\mathbf{I}_{HR}$  in such case.

On the other hand, if we increase weight  $\alpha$  to retain the input content, the resulting downsampled images will more likely resemble  $\mathcal{B}(\mathbf{I}_{HR})$  rather than the desired output  $(\mathbf{I}_{HR} * k)_{\downarrow 2}$ , as shown around edge and corner regions of Fig. 3(f)–(h). The tradeoff between preserving image contents and synthesizing an accurate distribution of the LR images occurs due to the inherent conflict between the predetermined downsampling operator  $\mathcal{R}_{HR}$  and the adversarial loss  $\mathcal{L}_{adv}$ . While the data term  $\mathcal{L}_{data}$  is necessary to learn an appropriate downsampling function, it also operates as a restrictive prior and prevents an accurate simulation of the target LR images. Therefore, an SR method developed with the biased downsampler may not perform well on the target distribution  $\mathcal{I}_{LR}$ , as conventional bicubic SR algorithms cannot be generalized on real-world images.

### 3.3 Data loss over low-frequency components

We propose an effective and generalizable formulation of the data term  $\mathcal{L}_{data}$  to address the limitations of the existing

approaches. Similar to the previous methods, our downsampler also takes an input HR image  $\mathbf{I}_{HR}$  and generates a downsampled image  $\mathbf{I}_{Down}$ . However, to preserve image contents and low-level structures in the downsampling process, we first define the operator  $\text{LPF}_m : \mathbb{R}^{H \times W} \rightarrow \mathbb{R}^{H/m \times W/m}$  as a combination of low-pass filtering and subsampling by  $m$ , which reduces the resolution of a given image by a factor of  $m > 1$ . Then, we rewrite the data loss in (2) with a low-frequency loss (LFL)  $\mathcal{L}_{data} = \mathcal{L}_{LFL}$  as follows:

$$\mathcal{L}_{LFL} = \|\text{LPF}_{ms}(\mathbf{I}_{HR}) - \text{LPF}_m(\mathbf{I}_{Down})\|_1, \quad (4)$$

where  $s$  is a scaling factor. Since the HR image  $\mathbf{I}_{HR}$  is  $s$  times larger than the downsampled one  $\mathbf{I}_{Down}$ , sizes of  $\text{LPF}_{ms}(\mathbf{I}_{HR})$  and  $\text{LPF}_m(\mathbf{I}_{Down})$  are the same. We adopt two different low-pass filters: the box and Gaussian, to formulate the loss term. As the HR and downsampled images have different resolutions, we adjust the filter weights proportionally so that the same context can be covered from the images with different scales. By default, we use  $32 \times 32$  and  $16 \times 16$  box filters for  $\text{LPF}_{ms}$  and  $\text{LPF}_m$ , respectively, with the scaling factor  $s = 2$ . We provide more details and ablations regarding the low-pass filters in Appendix A.

Fig. 4 illustrates the differences between the existing formulation and the proposed loss term. As shown in Fig. 4(a), the handcrafted operator constrains each pixel of the downsampled image  $\mathbf{I}_{Down}$  to be a predetermined function of the input HR image  $\mathbf{I}_{HR}$ . The primary limitation of such an approach is that the HR image  $\mathbf{I}_{HR}$  and operator  $\mathcal{R}_{HR}$  are both kept unchanged throughout the entire learning process. Therefore, the reference image  $\mathcal{R}_{HR}(\mathbf{I}_{HR})$  is also fixed for each HR image, which can bias the learning process. Thus, even with the adversarial training objective, the learned downsampler  $\mathcal{D}$  can be biased toward the predetermined operator  $\mathcal{R}_{HR}$  rather than the desired downsampling model, especially when the weight  $\alpha$  in (2) is large.

Our motivation is that we only need to preserve the low-frequency components of the image contents and structures. Fig. 4(b) demonstrate that the downsampled image  $\mathbf{I}_{Down}$  is no longer constrained to be a specific function of its HR counterpart with our LFL. Instead, we adopt a relaxed objective designed to match low-frequency structures between input and output of the downsampler. By doing so, the adversarial loss can play a significant role in rendering the unknown types of LR images. Our LFL is not a restrictive constraint for a general downsampling model and can be generalized well on various synthetic and real-world images. In other words, we can minimize the new data term  $\mathcal{L}_{LFL}$  without causing notable conflict with the adversarial loss for LR images  $\mathcal{I}_{LR}$  from an arbitrary downsampling model. More details are described in Section 4.5.

**Scale transfer learning.** The proposed LFL does not include any scale-specific formulation and can be generalized to larger scales, e.g.,  $\times 4$ . However, directly optimizing a high-scale downsampler may cause less stable behaviors due to the significant differences in the HR and downsampled images. To ensure stability, we learn a  $\times 2$  model  $\mathcal{D}^{\times 2}$  on the desired distribution  $\mathcal{I}_{LR}$  and repeat it  $n > 1$  times [19] to obtain the  $\times 2^n$  downsampling models  $\mathcal{D}^{\times 2^n}$  by following:

$$\mathcal{D}^{\times 2^n} = \mathcal{D}^{\times 2^{n-1}} \circ \mathcal{D}^{\times 2} \quad (n > 1). \quad (5)$$

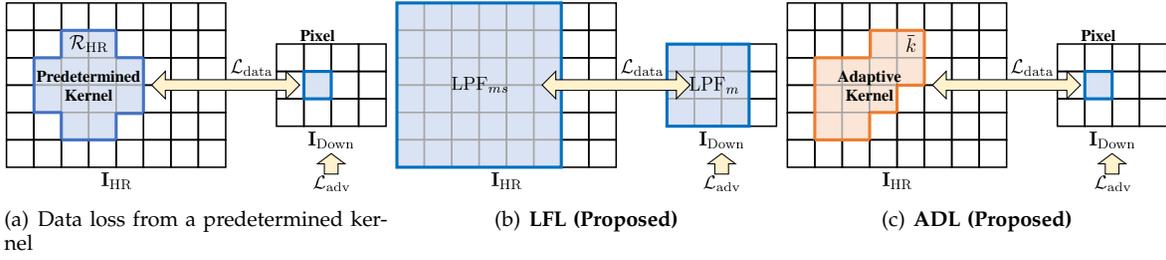


Fig. 4. **Different formulation for the data term.** We visualize how pixels in  $\mathbf{I}_{\text{Down}}$  is constrained to  $\mathbf{I}_{\text{HR}}$  depending on the data term  $\mathcal{L}_{\text{data}}$ . (a) Data loss from a predetermined kernel. (b) In the proposed LFL, we apply low-pass filters to HR and downsampled images so that image contents can be preserved across different scales regardless of the downsampling model. (c) In our adaptive data term, the orange kernel is learned from training samples and iteratively adjusted inside the training loops rather than handcrafted.

### 3.4 Adaptive data loss

Our LFL is designed to reduce the bias from selecting a predetermined downsampling operator for the data loss  $\mathcal{L}_{\text{data}}$ . While this formulation enables LFL to be generalized well across various unknown degradations, several limitations exist. For example, an inherent ambiguity in LFL makes it challenging to solve the optimization problem because the LR images from the different downsampling processes may share similar low-frequency components, i.e.,  $\text{LPF}_m(\mathbf{I}_a) = \text{LPF}_m(\mathbf{I}_b)$  for  $\mathbf{I}_a \neq \mathbf{I}_b$ . Considering that our goal is to simulate the unknown downsampling model  $\mathcal{D}^*$  with a CNN-based downsampler  $\mathcal{D}$ , the ideal data loss  $\mathcal{L}_{\text{data}}$  should be zero only if the condition  $\mathcal{D}(\mathbf{I}_{\text{HR}}) \equiv \mathcal{D}^*(\mathbf{I}_{\text{HR}})$  satisfies. The primary limitation of LFL is that it is designed to maintain consistency between HR and LR images, not to simulate structures of LR images in the target distribution. Since LFL only considers low-frequency components in the image, optimizing the term is an ill-posed problem where numerous possible  $\mathcal{D}$  exist. In particular, minimizing LFL allows the downsampler to generate valid LR images, while it is not guaranteed that the learned downsampler achieves our desired behavior. Therefore, the definition of LFL is generalizable but cannot be an optimal one for any arbitrary downsampling model due to the ambiguity.

Moreover, LFL can be problematic when the downsampled image  $\mathbf{I}_{\text{Down}}$  is corrupted with high-frequency noise, which is suppressed after low-pass filtering. Since the proposed LFL cannot reject noisy estimations, it is challenging to generate clean and accurate LR samples of the desired distribution. Consequently, the downsampler heavily relies on adversarial loss to simulate an accurate distribution of  $\mathcal{I}_{\text{LR}}$ , which may not be very stable in practice [58].

Therefore, we propose an adaptive data loss (ADL) to complement the limitations of LFL. The primary motivation is that the LFL-based downsampler  $\bar{\mathcal{D}}$  can serve as a dataset-specific objective if  $\bar{\mathcal{D}}$  and the ground-truth downsampling model  $\mathcal{D}^*$  are similar to some extent. To formulate the ADL, we first reduce the noise in the pre-trained model  $\bar{\mathcal{D}}$ . Rather than introducing a new objective term in (2) for regularization, we retrieve a low-rank approximation of the learned network with a simple function. From the observation that a proper downsampling function consists of low-pass filtering and decimation [17], [18], [19], [50], we linearize the learned downsampling model  $\bar{\mathcal{D}}$  to a corresponding 2D kernel  $\bar{k}$ :

$$\bar{k} = \underset{k}{\operatorname{argmin}} \sum_{i=1}^N \left\| (\mathbf{I}_{\text{HR}}^i * k)_{\downarrow s} - \bar{\mathcal{D}}(\mathbf{I}_{\text{HR}}^i) \right\|_2^2, \quad (6)$$

where  $\mathbf{I}_{\text{HR}}^i$  denotes an  $i$ -th example to estimate the kernel and  $N$  is the total number of samples that have been used, respectively. We note that there exists a closed-form solution for the least-squares in (6). Since (6) can be interpreted as an average of the possibly noisy downsampling network over  $N$  inputs, the kernel  $\bar{k}$  is a regularized representation of the pre-trained network  $\bar{\mathcal{D}}$ . With the estimated kernel  $\bar{k}$ , a novel ADL for data term  $\mathcal{L}_{\text{data}} = \mathcal{L}_{\text{ADL}}$  is defined as follows:

$$\mathcal{L}_{\text{ADL}} = \left\| (\mathbf{I}_{\text{HR}} * \bar{k})_{\downarrow s} - \mathbf{I}_{\text{Down}} \right\|_1. \quad (7)$$

While (7) looks identical to the data terms with handcrafted downsampling in (3), we can deduce several merits from the ADL formulation. Unlike the predetermined operators  $\mathcal{R}_{\text{HR}}$  or  $\mathcal{B}$ , the kernel  $\bar{k}$  is adaptively learned from the training data and shows less conflict to the adversarial loss  $\mathcal{L}_{\text{adv}}$ . In other words, the linear downsampling process in (7) is less likely to deviate considerably from our desired downsampling model  $\mathcal{D}^*$ . Compared with the LFL formulation, the ADL term can provide a stable training objective and prevent the downsampler from learning false-negative cases. Moreover, the learned downsampling model  $\mathcal{D}$  is not constrained to be a deep linear network [19], as we jointly optimize the adversarial loss  $\mathcal{L}_{\text{adv}}$  with the ADL.

Also, we introduce two modifications to utilize our ADL effectively in practice. First, the downsampler  $\mathcal{D}$  has been observed to simulate a target downsampling model  $\mathcal{D}^*$  to some extent under the LFL, even with few training iterations. Rather than using a fully pre-trained model  $\bar{\mathcal{D}}$  for the kernel estimation, we start from scratch and train the downsampler for  $t_{\text{warm-up}}$  iterations with the LFL. We then replace our data term with the ADL, in which the kernel  $\bar{k}$  is calculated from the downsampler  $\mathcal{D}$  after  $t_{\text{warm-up}}$  updates. Second, we periodically adjust the kernel  $\bar{k}$  to prevent our downsampling model from being biased toward a fixed operator. Similar to (3), our ADL may also bias the training process unless  $\mathcal{D}^*(\mathbf{I}_{\text{HR}}) \equiv (\mathbf{I}_{\text{HR}} * \bar{k})_{\downarrow s}$  holds. Thus, we periodically update the kernel  $\bar{k}$  by retrieving it from the currently learned downsampler. Even if the initial estimation  $\bar{k}$  is less accurate, such periodic updates allow the kernel to be adaptively adjusted during learning loops. The training pipeline of our downsampler  $\mathcal{D}$  with the modified ADL formulation is summarized in Algorithm 1.

### 3.5 Image super-resolution

To learn the SR model  $\mathcal{S}(\cdot; \phi)$ , we first generate the LR images  $\mathbf{I}_{\text{Down}}$  from HR images with the learned downsampler  $\mathcal{D}$  to construct a training set of  $(\mathbf{I}_{\text{Down}}, \mathbf{I}_{\text{HR}})$  pairs.

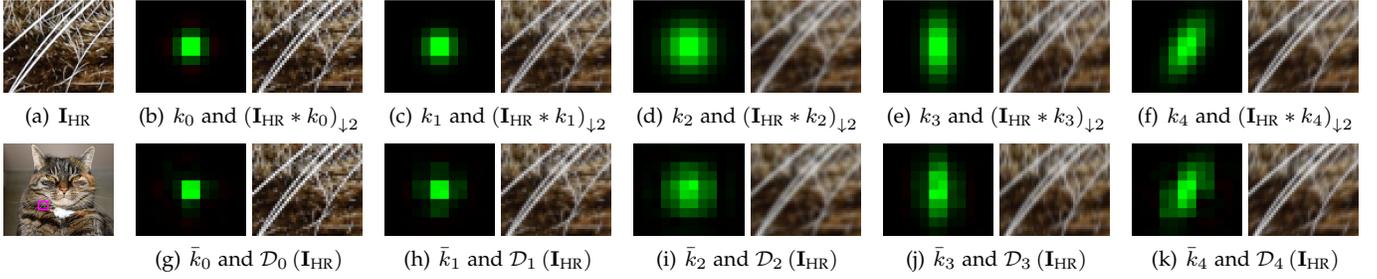


Fig. 5. **Examples of bicubic and randomly selected Gaussian kernels with corresponding  $\times 2$  LR images.** (a) We use DIV2K [7] ‘0869.png’ for the HR image  $\mathbf{I}_{\text{HR}}$ . (b)-(f) We note that the bicubic kernel  $k_0$  contains positive (green) and small negative (red) values together. The former two Gaussian kernels  $k_1$  and  $k_2$  are isotropic, while later kernels  $k_3$  and  $k_4$  are anisotropic. We note that there exist subtle differences between images from different downsampling kernels. (g)-(k) We also visualize downsampled images  $\mathbf{I}_{\text{Down}}$  from the proposed ADL formulation. Here,  $\mathcal{D}_i$  and  $\bar{k}_i$  refer to the downsampling CNN  $\mathcal{D}$  and approximated kernel  $\bar{k}$  in Algorithm 1 that are learned on the synthetic DIV2K dataset from  $k_i$ . Kernel boundaries are cropped for better visualization. Best viewed with digital zoom.

#### Algorithm 1 ADL for learning our downsampler $\mathcal{D}$

**Input:** Set of HR patches  $\mathcal{I}_{\text{HR}}$ , set of unpaired LR patches  $\mathcal{I}_{\text{LR}}$ , warm-up interval  $t_{\text{warm-up}}$ , update interval  $t_{\text{update}}$ , total training iterations  $T$ , and learning rate  $\eta$ .

**Output:** Downsampler parameters  $\theta_{\mathcal{D}}$  and discriminator parameters  $\theta_{\mathcal{F}}$ .

- 1:  $\theta_{\mathcal{D}}, \theta_{\mathcal{F}} \leftarrow \mathcal{N}(0, 0.02^2)$ . // Parameter initialization [58].
- 2:  $\bar{k} = \text{None}$ .
- 3: **for**  $i = 1 : T$  **do**
- 4:  $\mathbf{I}_{\text{LR}} \sim \mathcal{I}_{\text{LR}}, \mathbf{I}_{\text{HR}} \sim \mathcal{I}_{\text{HR}}$ . // Sample training batches.
- 5:  $\mathbf{I}_{\text{Down}} = \mathcal{D}(\mathbf{I}_{\text{HR}}; \theta_{\mathcal{D}})$ .
- 6:  $\theta_{\mathcal{F}} \leftarrow \theta_{\mathcal{F}} - \eta \nabla_{\theta_{\mathcal{F}}} \mathcal{L}_{\mathcal{F}}$ . // Update  $\theta_{\mathcal{F}}$  by (2).
- 7: **if**  $i < t_{\text{warm-up}}$  **then**
- 8: Calculate  $\mathcal{L}_{\text{data}}$  with (4).
- 9: **else**
- 10: **if**  $\text{mod}(i, t_{\text{update}}) == 0$  or  $\bar{k}$  is None **then**
- 11: Calculate  $\bar{k}$  from  $\mathcal{D}$ . // Retrieve the kernel.
- 12: **end if**
- 13: Calculate  $\mathcal{L}_{\text{data}} = \mathcal{L}_{\text{ADL}}$  with (7).
- 14: **end if**
- 15:  $\theta_{\mathcal{D}} \leftarrow \theta_{\mathcal{D}} - \eta \nabla_{\theta_{\mathcal{D}}} \mathcal{L}_{\mathcal{D}}$ . // Update  $\theta_{\mathcal{D}}$  by (2).
- 16: **end for**

A downsampling-specific SR model can be trained in a supervised manner by optimizing the  $L_1$  loss [5], [6]:

$$\begin{aligned} \mathcal{L}_{\mathcal{S}} &= \|\mathbf{I}_{\text{HR}} - \mathbf{I}_{\text{SR}}\|_1 \\ &= \|\mathbf{I}_{\text{HR}} - \mathcal{S}(\mathbf{I}_{\text{Down}})\|_1 = \|\mathbf{I}_{\text{HR}} - \mathcal{S}(\mathcal{D}(\mathbf{I}_{\text{HR}}))\|_1, \end{aligned} \quad (8)$$

where  $\mathbf{I}_{\text{SR}} = \mathcal{S}(\mathbf{I}_{\text{Down}})$  refers to a super-resolved image. As shown in (8), our approach does not require any paired examples, i.e., LR image  $\mathbf{I}_{\text{LR}}$ , to learn the SR model for  $\mathcal{I}_{\text{LR}}$ .

One of our contributions is that the downsampling and SR models can be learned independently. For instance, it is straightforward to introduce perceptual objective [1], [8], [42], [45] for the SR network, which can be used to reconstruct photo-realistic results. To reconstruct more realistic textures from the real-world LR images, we jointly optimize  $\mathcal{L}_{\mathcal{P}}$  and  $\mathcal{L}_{\mathcal{G}}$  to learn the perceptual SR model  $\mathcal{P}(\cdot; \theta_{\mathcal{P}})$  and the discriminator network  $\mathcal{G}(\cdot; \theta_{\mathcal{G}})$  respectively:

$$\begin{aligned} \mathcal{L}_{\mathcal{P}} &= \|\mathcal{V}_{54}(\mathbf{I}_{\text{HR}}) - \mathcal{V}_{54}(\mathbf{I}_{\text{SR-P}})\|_1 + \beta \mathcal{L}_{\text{adv-P}}, \\ \mathcal{L}_{\mathcal{G}} &= -\mathbb{E}[\log \mathcal{G}(\mathbf{I}_{\text{HR}})] - \mathbb{E}[\log(1 - \mathcal{G}(\mathbf{I}_{\text{SR-P}}))], \end{aligned} \quad (9)$$

where  $\mathcal{V}_{54}$  is features of the pre-trained VGG-19 [1], [8], [59] network after the `conv5_4` layer,  $\mathbf{I}_{\text{SR-P}} = \mathcal{P}(\mathbf{I}_{\text{LR}})$  is a super-

resolved image,  $\mathcal{L}_{\text{adv-P}} = -\mathbb{E}[\log \mathcal{G}(\mathbf{I}_{\text{SR}})]$  is adversarial loss, and  $\beta = 0.02$  is a hyperparameter, respectively.

## 4 EXPERIMENTS

We implement our method based on the PyTorch framework. More results can be further provided in our Appendix and project site: <https://cv.snu.ac.kr/research/ADL>. We will also release the source code and pre-trained models.

### 4.1 Experimental setups

**Dataset.** To validate whether our method can simulate an unknown distribution of LR images accurately, we construct a synthetic dataset by using a bicubic kernel ( $k_0$ ) and the Gaussian kernels ( $k_1$ - $k_4$ ) with random shapes [17], [18]. Then, we obtain LR inputs for the test from HR images by following (1). We visualize the different  $\times 2$  degradation kernels  $k_i$  used in our experiments and the corresponding LR images in Fig. 5. For the  $\times 4$  configurations, we use two times the enlarged versions of the  $\times 2$  kernels. More details about the selected kernels are described in Appendix B.

We construct unpaired training data by dividing 800 HR images from the DIV2K [7] training that is split by half. For each degradation kernel, we assign 400 HR samples (‘0001.png’-‘0400.png’) to  $\mathcal{I}_{\text{HR}}$ . The remaining 400 images (‘0401.png’-‘0800.png’) are used to synthesize LR samples and allocated to  $\mathcal{I}_{\text{LR}}$ . The images do not overlap between  $\mathcal{I}_{\text{HR}}$  and  $\mathcal{I}_{\text{LR}}$ . With the proposed LFL and ADL formulation, the downsampler  $\mathcal{D}$  can learn to simulate the distribution of LR samples  $\mathcal{I}_{\text{LR}}$  by using the given HR images  $\mathcal{I}_{\text{HR}}$ . For fair evaluations, we use another 100 images from the DIV2K [7] validation set to generate test inputs for different kernel  $k_i$ .

**Evaluation metrics.** We evaluate our downsampling methods in two aspects. As the primary goal of our methods is generating training examples to learn SR models on an unknown distribution of LR images, we generate pairs of  $(\mathbf{I}_{\text{Down}}, \mathbf{I}_{\text{HR}})$  using 400 HR images in  $\mathcal{I}_{\text{HR}}$  with the learned downsampler  $\mathcal{D}$  for each test degradation  $k_i$ . Then, we train the SR model as described in Section 3.5, and report the PSNR values between the reconstructed images  $\mathbf{I}_{\text{SR}}$  and the reference HR images  $\mathbf{I}_{\text{HR}}$ . We note that generating more accurate LR images allows the following SR model to improve generalization on the inputs from unknown degradation. In addition to the SR task, we also measure the PSNR values between the downsampled images  $\mathbf{I}_{\text{Down}}$  and ground-truth

TABLE 1

**Evaluation of LR images from our unsupervised downsampler.** We evaluate PSNR (dB) between downsampled and ground-truth LR images on the synthetic DIV2K dataset for each kernel  $k_i$ . The best and second-best methods are **bolded** and underlined, respectively.

$\mathcal{L}_{\text{data}}$	$s$	PSNR $^\dagger$ between $\mathbf{I}_{\text{LR}}$ and $\mathbf{I}_{\text{Down}}$				
		$k_0$	$k_1$	$k_2$	$k_3$	$k_4$
$\ \mathcal{B}(\mathbf{I}_{\text{HR}}) - \mathbf{I}_{\text{Down}}\ _1$	$\times 2$	40.70	39.64	35.79	37.88	37.22
$\ \text{AP}^s(\mathbf{I}_{\text{HR}}) - \mathbf{I}_{\text{Down}}\ _1$		40.33	38.54	36.03	37.35	35.13
$\mathcal{L}_{\text{LFL}}$ in (4) ( <b>Proposed</b> )		<u>43.16</u>	<u>42.03</u>	<u>43.30</u>	<u>43.69</u>	<u>43.75</u>
$\mathcal{L}_{\text{ADL}}$ in (7) ( <b>Proposed</b> )		<b>45.83</b>	<b>45.61</b>	<b>46.34</b>	<b>44.86</b>	<b>46.41</b>
$\ \mathcal{B}(\mathbf{I}_{\text{HR}}) - \mathbf{I}_{\text{Down}}\ _1$	$\times 4$	26.36	26.91	26.85	25.54	25.67
$\ \text{AP}^s(\mathbf{I}_{\text{HR}}) - \mathbf{I}_{\text{Down}}\ _1$		24.18	25.63	25.40	26.64	26.70
$\mathcal{L}_{\text{LFL}}$ in (4) ( <b>Proposed</b> )		<u>31.13</u>	<u>34.28</u>	<u>39.66</u>	<u>38.31</u>	<u>37.17</u>
$\mathcal{L}_{\text{ADL}}$ in (7) ( <b>Proposed</b> )		<b>38.24</b>	<b>38.12</b>	<b>43.54</b>	<b>39.08</b>	<b>41.10</b>

TABLE 2

### Training configurations of different SR methods.

All the other hyperparameters are kept fixed to train those SR models.

We note that the downsampler  $\mathcal{D}$  is learned for each specific degradation in an unsupervised manner.

Method	Training input	Training target
Bicubic	$\mathcal{B}(\mathbf{I}_{\text{HR}}) = (\mathbf{I}_{\text{HR}} * k_0)_{\downarrow s}$	$\mathbf{I}_{\text{HR}}$
Oracle	$(\mathbf{I}_{\text{HR}} * k_i)_{\downarrow s}$	
Proposed	$\mathcal{D}(\mathbf{I}_{\text{HR}})$	

LR images to quantitatively evaluate the performance of the learned downsampling models  $\mathcal{D}$ . All PSNR values are calculated using RGB channels rather than luminance.

**Model architecture.** We use the patch-based discriminator [60] with the instance normalization [61] for training. For the SR task, we use a small EDSR [6] model as the baseline with 1.5M parameters. To demonstrate that our method is orthogonal to the selection of the SR backbone, we also introduce a larger RRDB [8] architecture with 16.7M parameters. The details regarding our downsampling and discriminator CNNs are described in Appendix E.

**Hyperparameters.** In all experiments, we use a  $32 \times 32$  box filter for  $\text{LPF}_{m_s}$  and the one with  $16 \times 16$  spatial size for  $\text{LPF}_m$  with a scale factor of 2. The ablation studies about the filter selection and relevant hyperparameters are described in Section 4.5. In training, one epoch consists of 1,741 iterations, which is proportional to the number of total training patches. More details are provided in Appendix F.

## 4.2 Evaluating simulated LR images

The primary contribution of our LFL and ADL is that they do not make a conflict with the adversarial loss, which guides downsampled images to resemble LR samples from an unknown distribution. To demonstrate the advantages of the proposed framework when simulating an arbitrary downsampling process, we compare our LFL and ADL to data terms using predetermined operators. Maeda [22] proposes to utilize bicubic downsampled images in an unsupervised downsampling model, especially for cycle consistency and identity loss terms. While the unsupervised learning approach from Maeda [22] is not the same as our formulation, the objective between the generated LR and bicubic downsampled images can be interpreted as  $\mathcal{R}_{\text{HR}} = \mathcal{B}$  in (3). Similarly, Bulat *et al.* [24] used an  $s \times s$  average pooling ( $\text{AP}^s$ ) for the resizing operator  $\mathcal{R}_{\text{HR}}$  in the data term  $\mathcal{L}_{\text{data}}$ , where  $s$  corresponds to a scaling factor.

We note that direct comparisons between ours and the existing generation-based methods [22], [24], including Lugmayr *et al.* [23] are not conducted due to several reasons. First, we explicitly find the unknown downsampling operator, while previous approaches focus on modeling noise and artifacts in real-world LR images. In addition, as those methods do not provide source code, evaluation on diverse synthetic kernels cannot be carried out for fair comparisons. Therefore, we train multiple downsampling networks under different data terms  $\mathcal{L}_{\text{data}}$  on different synthetic kernels ( $k_0$ – $k_4$ ) and scales ( $\times 2$  and  $\times 4$ ). Then, we compare how the proposed data term outperforms the previous formulations in terms of the feasibility of the synthesized samples.

Table 1 illustrates the average PSNR between the generated LR images from each downsampler and ground-truth. When the predetermined operator is well-matched with a ground-truth downsampling function, e.g., using  $\mathcal{B}$  to estimate  $k_0$ , the unsupervised models effectively simulate target LR images. However, if the predetermined functions (bicubic and average pooling) are not overlapped with the unknown degradation kernel ( $k_1$ – $k_4$ ), the data term  $\mathcal{L}_{\text{data}}$  biases the training objective and conflicts with the adversarial loss. Table 1 demonstrates that the conflict affects the learned downsampling model in a negative way and makes the SR model less generalizable, even with synthetic kernels. On the other hand, the proposed LFL and ADL terms can be generalized better and facilitate the downsampler  $\mathcal{D}$  to generate accurate LR images for various configurations.

## 4.3 SR on the synthetic examples

Using generated LR images from our downsampler, we train baseline EDSR [6] and RRDB [8] and evaluate them on each degradation kernel  $k_i$  individually on three different configurations described in Table 2. In the bicubic configuration, bicubic-downsampled images are used to train the SR model as those in existing approaches [1], [4], [26]. We note that the bicubic models are shared across different setups. In contrast, our method first learns a degradation-specific downsampling model  $\mathcal{D}$  from unpaired LR and HR images and leverages it to generate training samples for the SR model. We also introduce an oracle for each degradation  $k_i$ , where the SR model can fully utilize the ground-truth kernel to synthesize training images. As the distributions of the training and test images are matched, the oracle serves as an upper bound for a specific degradation kernel  $k_i$ .

Table 3 compares various SR methods on the synthetic DIV2K dataset. EDSR and RRDB trained on bicubic LR images perform well when the inputs are also formed by the bicubic kernel ( $k_0$ ). However, they do not generalize well when the inputs are downsampled by different kernels ( $k_1$ – $k_4$ ), as the distribution of the test images deviates significantly from that of the training samples. Also, the larger RRDB network does not bring any advantages over the smaller EDSR model, showing that the bicubic SR models cannot be generalized to unknown types of LR images.

On the other hand, EDSR and RRDB achieve significant performance gains over the other approaches when the training LR images are generated from the proposed LFL and ADL. As discussed previously in Section 4.2, our approach can generate a set of *faithful* LR and HR training

TABLE 3  
Blind super-resolution results on synthetic LR images.

We show PSNR (dB) between ground-truth HR and SR images from various methods on the synthetic DIV2K test dataset. Performance is not reported (–) if the pre-trained model is available only for a specific scale or cannot generate output images.  $k_0$  refers to the bicubic kernel.

Method	PSNR <sup>†</sup> for $\times 2$ SR					PSNR <sup>†</sup> for $\times 4$ SR				
	$k_0$	$k_1$	$k_2$	$k_3$	$k_4$	$k_0$	$k_1$	$k_2$	$k_3$	$k_4$
EDSR [6] (Bicubic)	34.61	31.51	27.76	27.91	27.95	28.92	26.35	24.06	24.21	24.21
RRDB [8] (Bicubic)	–	–	–	–	–	29.45	26.44	24.07	24.22	24.22
EDSR (Oracle)	34.61	34.44	33.64	33.23	33.27	28.92	28.73	28.02	27.79	27.84
RRDB (Oracle)	–	–	–	–	–	29.45	29.28	28.39	28.08	28.62
KernelGAN [19] + ZSSR [20]	22.32	26.42	30.44	29.10	29.12	20.11	24.67	25.85	25.21	25.36
IKC [17]	–	–	–	–	–	<b>28.59</b>	<u>28.07</u>	<b>27.65</b>	24.15	25.12
BlindSR [18]	26.56	–	26.62	26.49	–	–	–	–	–	–
LFL + EDSR ( <b>Proposed</b> )	<u>33.91</u>	<u>33.26</u>	<u>31.38</u>	<u>31.48</u>	<u>31.57</u>	27.45	27.31	26.69	26.65	26.33
ADL + EDSR ( <b>Proposed</b> )	<b>34.07</b>	<b>33.68</b>	<b>32.51</b>	<b>32.08</b>	<b>32.05</b>	28.16	28.04	27.08	<u>26.82</u>	<u>26.97</u>
ADL + RRDB ( <b>Proposed</b> )	–	–	–	–	–	<u>28.55</u>	<b>28.49</b>	<u>27.51</u>	<b>27.00</b>	<b>27.19</b>

TABLE 4  
Blind super-resolution results on realistic LR images.

We provide PSNR (dB) between ground-truth HR and SR results from different methods on the RealSR-V3 [14] dataset. Since the KernelGAN [19] and ZSSR [20] combination is learned on *each* test image, they require  $\times 50$  parameters in practice to handle 50 inputs.

Method	# Parameters	Training data	PSNR <sup>†</sup> for $\times 2$ SR		PSNR <sup>†</sup> for $\times 4$ SR	
			Canon	Nikon	Canon	Nikon
EDSR [6] (Bicubic)	1.5M	Synthetic	30.58	30.00	26.05	25.89
RRDB [8] (Bicubic)	16.7M	(Bicubic $k$ )	–	–	26.05	25.91
EDSR (Oracle)	1.5M	RealSR-V3	32.45	31.59	27.59	27.14
RRDB (Oracle)	16.7M	(Paired)	–	–	27.90	27.39
KernelGAN [19] + ZSSR [20]	$50 \times (0.2M + 0.2M)$	A given $\mathbf{I}_{LR}$	28.79	27.54	23.68	22.46
IKC [17]	9.0M	Synthetic	–	–	25.71	25.27
BlindSR [18]	1.1M	(Multiple $k$ )	25.80	24.17	–	–
LFL + EDSR ( <b>Proposed</b> )	0.9M + 1.5M	RealSR-V3 (Unpaired)	<u>31.67</u>	<u>30.75</u>	26.47	25.90
ADL + EDSR ( <b>Proposed</b> )	0.9M + 1.5M		<b>31.81</b>	<b>30.99</b>	<u>26.79</u>	<u>26.46</u>
ADL + RRDB ( <b>Proposed</b> )	0.9M + 16.7M		–	–	<b>26.90</b>	<b>26.64</b>

pairs so that the following SR models can achieve much better performance on LR images from some unknown downsampling process. Since LFL does not bias the learned downsampler  $\mathcal{D}$  to a specific downsampling operator, e.g., bicubic or average pooling, the respective EDSR and RRDB generalize well across various kernels ( $k_0$ – $k_4$ ) and scales ( $\times 2$  and  $\times 4$ ). Furthermore, our downsampling model with ADL generates more accurate training LR images for the SR models and brings significant improvements to EDSR and RRDB across all kernel configurations consistently.

Interestingly, a larger RRDB model with ADL achieves better performance and even comparable to the oracle EDSR, especially on the  $\times 4$  SR task with  $k_0$  and  $k_1$ . If the distribution of the downsampled images, i.e.,  $\mathcal{D}(\mathbf{I}_{HR})$ , deviates much from that of the target LR images, then a better fitting to the training data may worsen the performance on the test images. Therefore, the performance gain of the RRDB model demonstrates that our unsupervised downsampling framework can faithfully simulate the distribution of the target LR images to a certain extent. We also note that our data generation process is orthogonal to the architecture of the SR models. Therefore, integrating the proposed downsampling models with state-of-the-art SR architectures [9], [10] can directly improve performance.

We also apply the existing approaches to various synthetic degradation kernels. The combination of KernelGAN [19] and ZSSR [20] first estimates an input-specific

degradation kernel from a single image [19] and applies the zero-shot SR model [20] to deal with the arbitrary LR images. Compared with bicubic EDSR and RRDB, the single image approach achieves better performance on  $k_2$ – $k_4$ , demonstrating the importance of estimating image-specific kernel modeling for the blind SR task. However, this method does not perform well when test images are downsampled by  $k_0$  and  $k_1$ , as depicted by the significant decrease in PSNR with respect to the oracle model.

Instead of synthesizing realistic LR images, IKC [17] and BlindSR [18] utilize large-scale synthetic data in which the degradation kernels follow specific shapes. They first predict the kernel used to generate the given LR image and derive input-dependent SR results within a single network architecture. As the IKC model only considers isotropic Gaussian kernels, it achieves comparable performance to the oracle models on  $k_0$ – $k_2$ . However, it does not perform well on the  $k_3$  and  $k_4$  cases, where the degradation kernels are anisotropic. While BlindSR [18] also takes a similar strategy, it is less stable and unable to handle some inputs from  $k_1$  and  $k_4$ , and it also diverges. As shown in Fig. 6 (first row), RRDB that has learned on our training data can reconstruct realistic details from the challenging  $\times 4$  SR task.

#### 4.4 SR on the RealSR-V3 dataset

Our approach can also be applied to real-world LR images from unknown camera processing pipelines. For the quan-

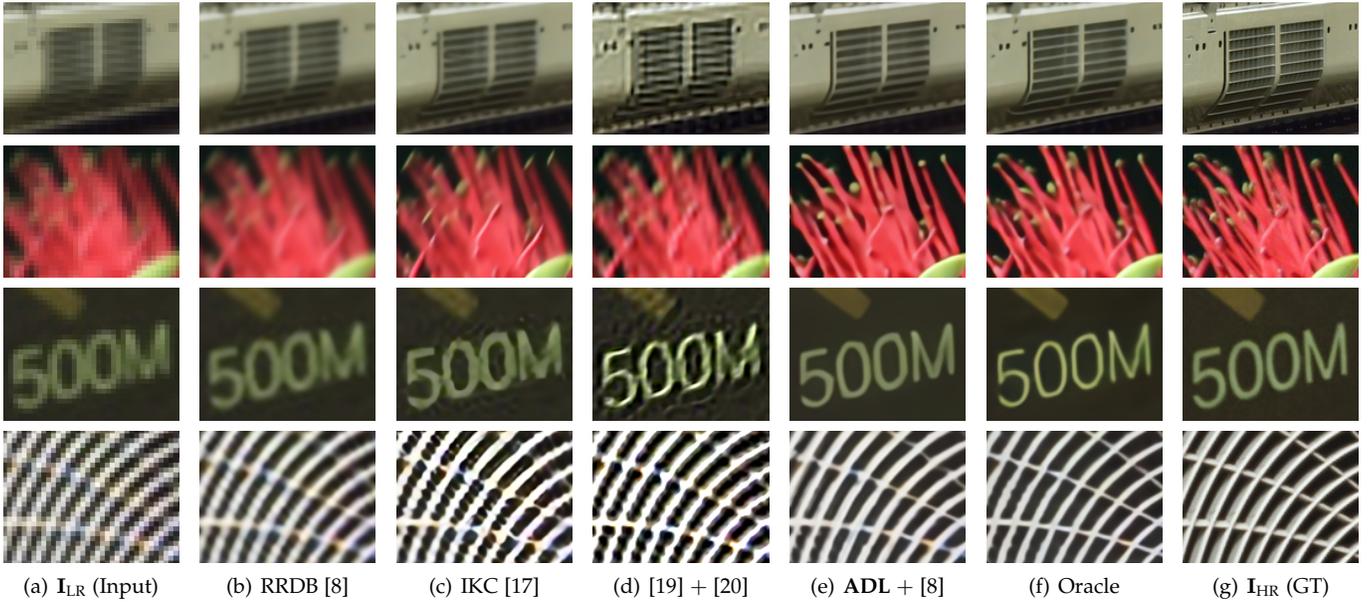


Fig. 6. **Qualitative  $\times 4$  SR results on the various datasets.** Patches in each row are from the synthetic DIV2K [7] dataset ‘0820.png ( $k_1$ ),’ ‘0853.png ( $k_4$ ),’ RealSR-V3 [14] dataset ‘Canon/006.png,’ and ‘Nikon/041.png,’ respectively. The RRDB [8] model is used as a backbone SR architecture for our ADL as well as the Oracle.

titative evaluation, we utilize RealSR-V3 [14] containing 200 well-aligned LR and HR image pairs for two different real-world cameras: Canon and Nikon. Similar to the description in Section 4.1, we divide the dataset by half for each camera model. The same amounts of images are assigned to  $\mathcal{I}_{HR}$  and  $\mathcal{I}_{LR}$  without overlapping. We learn the corresponding downsampling and SR models by following our pipeline, as described in Section 4.1. As the dataset provides accurately aligned LR and HR examples to train a supervised SR model, the oracle models learn from those image pairs.

Compared to the experiments on synthetic images in Section 4.3, real-world cases are more challenging. First, a set of LR images may share the similar but not exactly the same degradation process. Also, since the dataset mainly consists of indoor scenes and static objects without large motions, training the images may lack the diversity that can hinder generalization. Table 4 shows the results of the evaluated SR algorithms on RealSR-V3, where each of the Canon and Nikon split contains 50 test images. KernelGAN + ZSSR, IKC, and BlindSR do not perform well even compared to EDSR and RRDB learned on bicubic downsampled images. The primary reason is that numerous constraints in these methods, e.g., Gaussian kernels [17], [18] or kernel shape priors [19], do not usually hold for real-world scenes.

In contrast, our downsampling method with different SR backbones (LFL + EDSR, ADL + EDSR, and ADL + RRDB) achieve better results on both cameras at different scales ( $\times 2$  and  $\times 4$ ), compared with the other approaches. Even if the LR images in RealSR-V3 are not formulated from the same kernel, our LFL and ADL can learn an average of all possible downsampling operators and generalize well on the real-world dataset. We also demonstrate that the larger RRDB model performs better in the realistic case, showing that a better fitting on the generated LR images can help generalization on unseen real-world examples. Fig. 6 shows that our approach can reconstruct more visually pleasing results than the existing methods on RealSR-V3.

TABLE 5

**Ablation studies on the proposed method.**

We report how different training configurations for the downsampling network affect the SR results on the synthetic DIV2K dataset.

(a) Effect of the balancing parameter  $\alpha$  in (2).

Method \ $\alpha$	PSNR $^\dagger$ for $\times 2$ SR ( $k_4$ )			
	1	10	100	200
LFL + EDSR ( <b>Proposed</b> )	29.36	30.70	<u>31.49</u>	<b>31.57</b>
ADL + EDSR ( <b>Proposed</b> )	29.08	31.42	<b>32.05</b>	<u>32.02</u>

(b) Effect of the number of training LR images  $|\mathcal{I}_{LR}|$ .

Method \ $ \mathcal{I}_{LR} $	PSNR $^\dagger$ for $\times 2$ SR ( $k_4$ )			
	1	10	50	100
KernelGAN + ZSSR	29.12	–	–	–
LFL + EDSR ( <b>Proposed</b> )	25.43	28.70	29.86	31.21
ADL + EDSR ( <b>Proposed</b> )	29.84	30.20	31.36	32.10

(c) Effect of the joint training.

Method \ Configuration	PSNR $^\dagger$ (dB) for $\times 2$ SR ( $k_4$ )			
	Joint	+BP	+BP+FQ	Two-stage
ADL + EDSR ( <b>Proposed</b> )	<b>32.09</b>	31.39	31.50	<u>32.05</u>

#### 4.5 Ablation study

To see the contribution of each design component in our LFL and ADL, we conduct extensive ablation studies in this section. The selected hyperparameters are used throughout the entire experiments in Section 4.3, 4.4, and 4.7, without any additional adjustments. We note that an ablation regarding the shape of  $LPF_m$  is described in our Appendix A. The stability of our LFL and ADL is described in Appendix C.

**Effect of the balancing hyperparameter.** As we describe in Section 3.2, the predetermined downsampling operator  $\mathcal{R}_{HR}$  may bias the overall training objective of the downsampler  $\mathcal{L}_D$ , especially when the balancing hyperparameter  $\alpha$  is large. To validate that our LFL and ADL terms do not impose negative bias in the learning stage, we analyze how

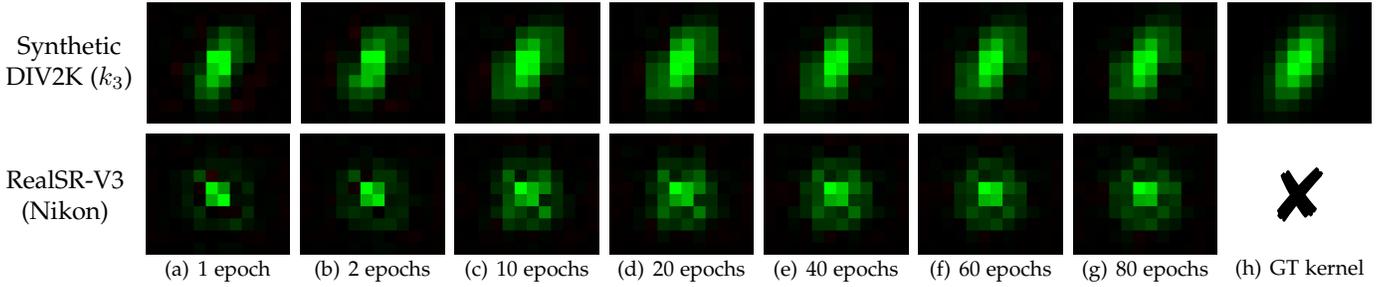


Fig. 7. **Evolution of the retrieved  $\times 2$  degradation kernel  $\bar{k}$  in the proposed ADL.** We visualize the estimated kernels from (6) on two different datasets. For simplicity, we refer 1,741 iterations as one epoch. Since we apply the ADL after 10 warm-up epochs, downsampling networks in (a) and (b) are trained under LFL, not ADL. Furthermore, (a) and (b) visualize the linear approximations of the learned downsamplers after a certain number of training epochs rather than the approximated kernels  $k$ . In the RealSR-V3 [14] dataset, no ground-truth kernel is available for the Nikon camera configuration. We crop image boundaries for better illustration.

the balance between data and adversarial losses in (2) affects the associated SR models. Table 5(a) shows that the SR model with LFL and ADL do not perform well when  $\alpha = 1$  or 10, as preserving image content is challenging during the downsampling process. When using relative larger values of  $\alpha$ , i.e.,  $\alpha = 100$  or 200, the baseline SR models with LFL and ADL terms perform reasonably well without making bias. As such, we choose  $\alpha = 200$  for the LFL and  $\alpha = 100$  for the ADL to achieve the best performance.

**Effect of the number of training samples.** In Section 4.3, we train the proposed downsampling model on 400 LR images generated from the *same* kernel  $k_i$ . Compared with the KernelGAN [19] method, which predicts a proper degradation kernel from a single input image, our approach requires more examples to estimate an unknown degradation accurately. For a fair comparison, we vary the number of LR images to train the downsampling model and analyze the performance of the following SR models. We use the first 1, 10, 50, and 100 examples from  $\mathcal{I}_{LR}$ , e.g., ‘0401.png’ for the  $|\mathcal{I}_{LR}| = 1$  case, to learn our downsampling model. The other hyperparameters are fixed unless mentioned otherwise.

Table 5(b) shows how the size of  $\mathcal{I}_{LR}$  for the downsampler affects the following SR performance. Our methods (LFL + EDSR and ADL + EDSR) gradually achieve better performance as the number of training samples increases, validating the effectiveness of using large-scale datasets. Nevertheless, even with a single LR sample, ADL + EDSR outperforms the single-image method. Table 5(b) also shows that ADL consistently outperforms LFL, especially when the number of training LR images is limited. Specifically, ADL + EDSR with a single LR image performs equally well as LFL + EDSR with 50 LR images. In Appendix D, we also analyze some opposite cases for fair comparison where KernelGAN is trained with multiple LR and HR images.

**Joint training of downsampling and SR networks.** Our two-stage (downsampling + SR) pipeline has several advantages. First, if the two models are jointly learned, one may affect the other to be suboptimal solution. For example, the downsampler may generate LR images that can be easily upsampled rather than accurately simulating the desired target. In addition, connecting the two models increases the algorithmic complexity, making hard to train the whole model. Finally, the two-stage approach accommodates more effective models and objective functions, as we have a fixed downsampling network and corresponding LR images.

On the other hand, it is possible to optimize the down-

sampling and SR networks jointly. Table 5(c) provides experimental results of joint training with different setups. To reduce the training time, we optimize downsampling and SR networks together (joint), contrary to the original formulation (two-stage). We note that the gradient from the SR model does not backpropagate to the downsampler in this configuration. Interestingly, the joint training approach achieves marginal performance gain to the SR network. Since we do not fix input of the SR network and keep updating the downsampler, it has similar effects to data augmentation and slightly improves the following SR model.

We also train the downsampling and SR networks together in an end-to-end manner, where backpropagated gradient from the SR model flows to the downsampler (+BP). However, this approach negatively affects the following SR network in two specific aspects. First, the downsampler tends to generate images that are easy to be upsampled rather than accurately simulating samples in the target LR distribution. Second, each color pixel in the generated image is a continuous variable, while pixels in our test samples only have 256 discontinuous values. We further introduce fake quantization (+BP+FQ) to deal with the second issue, where output of the downsampler are quantized while the gradient flows just as the pixel values are continuous. Although it brings +0.11dB performance gain, the end-to-end learning does not bring any advantage in our framework. As such, we use the two-stage approach in all the experiments.

#### 4.6 Analysis on ADL

As the estimated kernel  $\bar{k}$  in Algorithm 1 is derived from the training dataset, the ADL does not make a significant conflict with the adversarial loss  $\mathcal{L}_{adv}$ . To demonstrate the effectiveness of our adaptive adjustment strategy, we visualize how the estimated kernels  $\bar{k}$  on the synthetic DIV2K and RealSR-V3 datasets are updated in Fig. 7. We note that the kernels from the RealSR-V3 [14] dataset do not appear to be standard Gaussian forms that are preferred in the existing approaches [17], [18]. Since we do not constrain the downsampling network to resemble specific shapes of kernels, our approach can yield better generalizability.

In the synthetic cases, the estimated kernel  $\bar{k}$  should be similar to the ground-truth  $k_i$  for the following SR training. Thus, we show the kernel similarity [62] between the retrieved and ground-truth degradation kernels in Table 6(a) to validate that our prediction becomes more accurate as the training proceeds. It is also demonstrated that the more

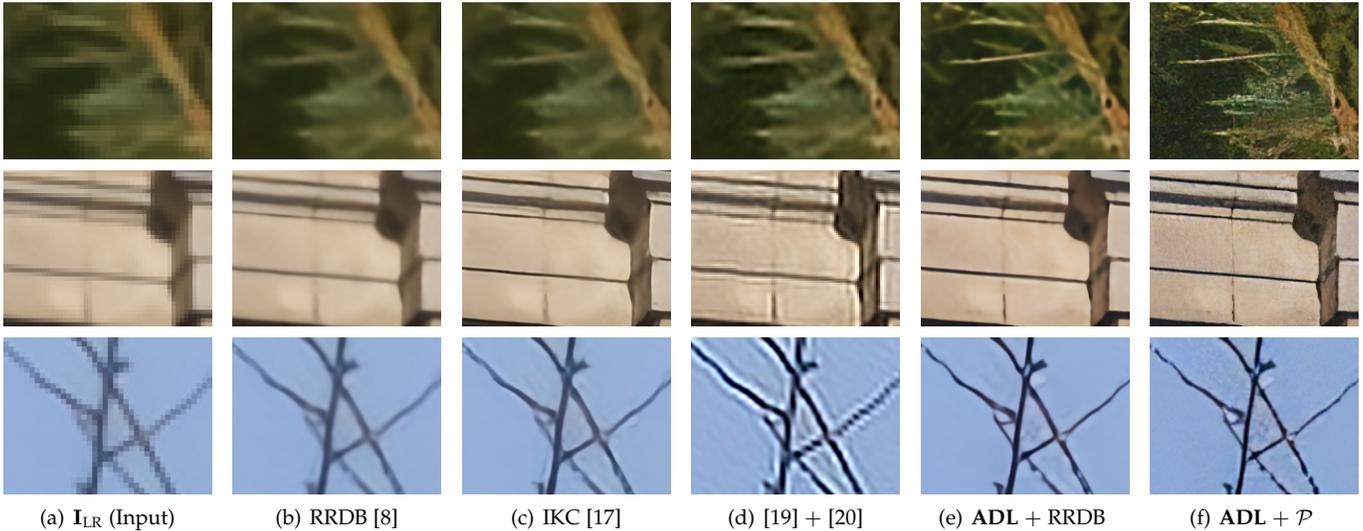


Fig. 8. **Perceptual  $\times 4$  SR results on the DPED dataset.** We note that no ground-truth HR images exist for the dataset. Therefore, quantitative comparison on the DPED images nor the oracle model is not available.  $\mathcal{P}$  denotes the RRDB [8] model learned with (9) to reconstruct more realistic textures and sharper details. From the top, patches are cropped from the DPED-val ‘20.png’, ‘49.png’, and ‘63.png’, respectively.

TABLE 6

**Ablation study about the proposed ADL method.**

To evaluate the SR performance, we use ADL + EDSR configuration and the synthetic DIV2K dataset. (a) We note that the kernel similarity [62] is measured after the last update, and  $\bar{k}$  is a linear approximation of the learned downsampler for each configuration. (b)  $\infty$  means that the kernel  $\bar{k}$  is not updated after the first estimation.

(a) Effect of the total training epochs in Algorithm 1.

Total training epochs $T$	20	40	60	80
Similarity $^\dagger$ ( $\bar{k}, k_4$ )	0.9682	0.9707	<u>0.9714</u>	<b>0.9718</b>
PSNR $^\dagger$ for $\times 2$ SR ( $k_4$ )	31.48	<u>31.96</u>	31.93	<b>32.05</b>

(b) Effect of the iterative adjustment in Algorithm 1.

Update interval $t$ (epochs)	ADL + EDSR ( <b>Proposed</b> )			
	1	10	20	$\infty$
PSNR $^\dagger$ for $\times 2$ SR ( $k_4$ )	31.65	<b>32.05</b>	<u>32.01</u>	31.85

(c) Effect of the number of samples for kernel estimation in (6).

The number of samples $N$	ADL + EDSR ( <b>Proposed</b> )			
	1	5	10	50
PSNR $^\dagger$ for $\times 2$ SR ( $k_4$ )	31.09	31.84	<u>32.01</u>	<b>32.05</b>

precise estimation supports the following SR network to perform better, and the performance is maximized at 80 epochs. Furthermore, Table 6(b) shows that our iterative update prevents the downsampler from being biased toward the fixed kernel  $\bar{k}$  and helps with the convergence. As we describe in Section 3.4, ADL is designed to stabilize the potentially noisy downsampling model. Therefore, the number of samples  $N$  for the kernel estimation is also crucial. Table 6(c) demonstrates that the ADL does not perform well only with a single training image. However, as the number of examples increases, our method can assist the following SR model to generate better results.

#### 4.7 SR on the real-world images

Our methods assume that the set of available LR images follow the same downsampling process. In practice, acquiring multiple images from a similar degradation pipeline, such

as photos from a fixed camera configuration [63] or multiple frames in a video, is relatively easier than collecting real-world LR and HR pairs. We validate the proposed method by using the DPED [63] dataset, which consists of low-quality photos captured by an iPhone 3GS camera. To train the downsampler, we assign random 120 images from the DPED [63] dataset as  $\mathcal{I}_{LR}$ , while DIV2K is used for the HR samples  $\mathcal{I}_{HR}$ . Since the dataset consists of low-quality examples captured by the iPhone 3GS camera, we remove the unknown noise and artifacts in  $\mathcal{I}_{LR}$  by applying the off-the-shelf RL-restore [64] algorithm as a preprocessing. Fig. 8 compares the SR results of the preprocessed DPED [63] dataset. As no ground-truth HR images exist in this dataset, we note that no oracle model is available for the dataset. Compared with the existing methods, our ADL facilitates RRDB [8] to reconstruct sharper edges and more detailed textures without introducing visual artifacts. Moreover, we train the ADL-based downsampler without RL-restore [64].

We also train the perceptual SR network  $\mathcal{P}$  using the images from ADL-based downsampler to demonstrate the merit of our approach for real-world SR. Compared with the PSNR-based model (ADL + RRDB) trained with (8), the perceptual RRDB model (ADL +  $\mathcal{P}$ ) from (9) reconstructs more realistic and visually pleasing results. The advantage of the two-stage approach is that we do not require additional training of the downsampler to introduce different optimization objectives for the SR network. Thus, the only difference between Fig. 8(e) and Fig. 8(f) is training loss while the backbone networks are the same. More additional qualitative SR results are presented in Appendix G.

In Fig. 9, we analyze how much our ADL is affected by noise and artifacts. For comparison, we evaluate the pretrained RealSR [56] and BSRGAN [49] models on the same DPED images. Since those methods explicitly consider noise and artifacts in LR inputs, their results look robust to such degradation. In contrast, our ADL yields sharper but a bit noisy outputs. Note that when the preprocessing is not applied (ADL-n), it gives blurry SR results since noisy LR samples prevent the discriminator from learning distinguishable features from downsampled images.

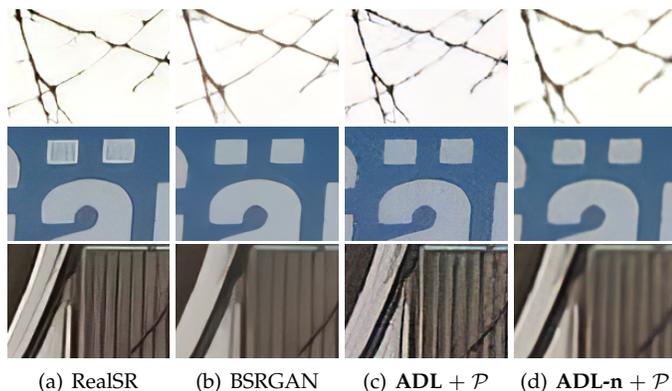


Fig. 9. **Effect of preprocessing on the DPED dataset.** In ADL, appropriate preprocessing is required for more effective learning. Patches are cropped from DPED-val ‘45.png,’ ‘63.png,’ and ‘84.png,’ respectively.

## 5 CONCLUSIONS

We propose a novel unsupervised method to estimate an unknown distribution of LR images using unpaired LR and HR examples. The proposed LFL and ADL terms facilitate the downsampler to accurately synthesize the LR images with the desired distribution. Compared to conventional approaches, we do not pose restrictive priors to the learned function in the adversarial training framework. Consequently, the existing SR models can be trained with our LR images and achieve significant performance gains on synthetic and realistic datasets. We also demonstrate that our approach can be applied to a set of arbitrary images [14], [63] in the wild. The results verify that the proposed method can be used to handle real-world SR problems. In the future work, we will extend our approach to estimating a feasible downsampling model with real-world noise jointly.

## ACKNOWLEDGEMENT

This work was partly supported by IITP grant funded by the Korea government [No. 2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)].

## REFERENCES

- [1] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, “Photo-realistic single image super-resolution using a generative adversarial network,” in *CVPR*, 2017.
- [2] Y. Bahat and T. Michaeli, “Explorable super resolution,” in *CVPR*, 2020.
- [3] B. Wronski, I. Garcia-Dorado, M. Ernst, D. Kelly, M. Krainin, C.-K. Liang, M. Levoy, and P. Milanfar, “Handheld multi-frame super-resolution,” *ACM TOG*, vol. 38, no. 4, pp. 1–18, 2019.
- [4] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” *TPAMI*, 2016.
- [5] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, “Deep laplacian pyramid networks for fast and accurate super-resolution,” in *CVPR*, 2017.
- [6] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, “Enhanced deep residual networks for single image super-resolution,” in *CVPR Workshops*, 2017.
- [7] E. Agustsson and R. Timofte, “NTIRE 2017 challenge on single image super-resolution: Dataset and study,” in *CVPR Workshops*, 2017.
- [8] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, “ESRGAN: enhanced super-resolution generative adversarial networks,” in *ECCV Workshops*, 2018.
- [9] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, “Image super-resolution using very deep residual channel attention networks,” in *ECCV*, 2018.
- [10] M. Haris, G. Shakhnarovich, and N. Ukita, “Deep back-projection networks for super-resolution,” in *CVPR*, 2018.
- [11] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, “Residual dense network for image super-resolution,” in *CVPR*, 2018.
- [12] X. Zhang, Q. Chen, R. Ng, and V. Koltun, “Zoom to learn, learn to zoom,” in *CVPR*, 2019.
- [13] C. Chen, Z. Xiong, X. Tian, Z.-J. Zha, and F. Wu, “Camera lens super-resolution,” in *CVPR*, 2019.
- [14] J. Cai, H. Zeng, H. Yong, Z. Cao, and L. Zhang, “Toward real-world single image super-resolution: A new benchmark and a new model,” in *ICCV*, 2019.
- [15] P. Wei, Z. Xie, H. Lu, Z. Zhan, Q. Ye, W. Zuo, and L. Lin, “Component divide-and-conquer for real-world image super-resolution,” in *ECCV*, 2020.
- [16] K. Zhang, W. Zuo, and L. Zhang, “Learning a single convolutional super-resolution network for multiple degradations,” in *CVPR*, 2018.
- [17] J. Gu, H. Lu, W. Zuo, and C. Dong, “Blind super-resolution with iterative kernel correction,” in *CVPR*, 2019.
- [18] V. Cornillère, A. Djelouah, W. Yifan, O. Sorkine-Hornung, and C. Schroers, “Blind image super-resolution with spatially variant degradations,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–13, 2019.
- [19] S. Bell-Kligler, A. Shocher, and M. Irani, “Blind super-resolution kernel estimation using an internal-gan,” in *NeurIPS*, 2019.
- [20] A. Shocher, N. Cohen, and M. Irani, ““Zero-Shot” super-resolution using deep internal learning,” in *CVPR*, 2018.
- [21] T. Zhao, W. Ren, C. Zhang, D. Ren, and Q. Hu, “Unsupervised degradation learning for single image super-resolution,” *arXiv*, 2018.
- [22] S. Maeda, “Unpaired image super-resolution using pseudo-supervision,” in *CVPR*, 2020.
- [23] A. Lugmayr, M. Danelljan, and R. Timofte, “Unsupervised learning for real-world super-resolution,” *arXiv*, 2019.
- [24] A. Bulat, J. Yang, and G. Tzimiropoulos, “To learn image super-resolution, use a GAN to learn how to do image degradation first,” in *ECCV*, 2018.
- [25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *NIPS*, 2014.
- [26] J. Kim, J. K. Lee, and K. M. Lee, “Accurate image super-resolution using very deep convolutional networks,” in *CVPR*, 2016.
- [27] N. Ahn, B. Kang, and K.-A. Sohn, “Fast, accurate, and lightweight super-resolution with cascading residual network,” in *ECCV*, 2018.
- [28] J. Li, F. Fang, K. Mei, and G. Zhang, “Multi-scale residual network for image super-resolution,” in *ECCV*, 2018.
- [29] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network,” in *CVPR*, 2016.
- [30] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, “Fast and accurate image super-resolution with deep laplacian pyramid networks,” *TPAMI*, vol. 41, no. 11, pp. 2599–2613, 2018.
- [31] Y. Wang, F. Perazzi, B. McWilliams, A. Sorkine-Hornung, O. Sorkine-Hornung, and C. Schroers, “A fully progressive approach to single-image super-resolution,” in *CVPRW*, 2018.
- [32] T. Tong, G. Li, X. Liu, and Q. Gao, “Image super-resolution using dense skip connections,” in *ICCV*, 2017.
- [33] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, “Residual dense network for image restoration,” *TPAMI*, 2020.
- [34] J. Kim, J. K. Lee, and K. M. Lee, “Deeply-recursive convolutional network for image super-resolution,” in *CVPR*, 2016.
- [35] Y. Qiu, R. Wang, D. Tao, and J. Cheng, “Embedded block residual network: A recursive restoration model for single-image super-resolution,” in *ICCV*, 2019.
- [36] Z. Li, J. Yang, Z. Liu, X. Yang, G. Jeon, and W. Wu, “Feedback network for image super-resolution,” in *CVPR*, 2019.
- [37] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang, “Second-order attention network for single image super-resolution,” in *CVPR*, 2019.
- [38] B. Niu, W. Wen, W. Ren, X. Zhang, L. Yang, S. Wang, K. Zhang, X. Cao, and H. Shen, “Single image super-resolution via a holistic attention network,” in *ECCV*, 2020.
- [39] Z. Hui, X. Wang, and X. Gao, “Fast and accurate single image super-resolution via information distillation network,” in *CVPR*, 2018.

- [40] W. Lee, J. Lee, D. Kim, and B. Ham, "Learning with privileged information for efficient image super-resolution," in *ECCV*, 2020.
- [41] X. Luo, Y. Xie, Y. Zhang, Y. Qu, C. Li, and Y. Fu, "LatticeNet: Towards lightweight image super-resolution with lattice block," in *ECCV*, 2020.
- [42] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *ECCV*, 2016.
- [43] M. S. M. Sajjadi, B. Scholkopf, and M. Hirsch, "EnhanceNet: Single image super-resolution through automated texture synthesis," in *ICCV*, 2017.
- [44] R. Mechrez, I. Talmi, and L. Zelnik-Manor, "The contextual loss for image transformation with non-aligned data," in *ECCV*, 2018.
- [45] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018.
- [46] W. Zhang, Y. Liu, C. Dong, and Y. Qiao, "RankSRGAN: Generative adversarial networks with ranker for image super-resolution," in *ICCV*, 2019.
- [47] M. S. Rad, B. Bozorgtabar, U.-V. Marti, M. Basler, H. K. Ekenel, and J.-P. Thiran, "SROBB: Targeted perceptual loss for single image super-resolution," in *ICCV*, 2019.
- [48] S. A. Hussein, T. Tirer, and R. Giryes, "Correction filter for single image super-resolution: Robustifying off-the-shelf deep super-resolvers," in *CVPR*, 2020.
- [49] K. Zhang, J. Liang, L. Van Gool, and R. Timofte, "Designing a practical degradation model for deep blind image super-resolution," *arXiv*, 2021.
- [50] T. Michaeli and M. Irani, "Nonparametric blind super-resolution," in *CVPR*, 2013.
- [51] J. Pan, Z. Hu, Z. Su, and M.-H. Yang, "Deblurring text images via l0-regularized intensity and gradient prior," in *CVPR*, 2014.
- [52] K. Zhang, L. Van Gool, and R. Timofte, "Deep unfolding network for image super-resolution," in *CVPR*, 2020.
- [53] Y.-S. Xu, S.-Y. R. Tseng, Y. Tseng, H.-K. Kuo, and Y.-M. Tsai, "Unified dynamic convolutional network for super-resolution with variational degradations," in *CVPR*, 2020.
- [54] R. Zhou and S. Susstrunk, "Kernel modeling super-resolution on real low-resolution images," in *ICCV*, 2019.
- [55] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017.
- [56] X. Ji, Y. Cao, Y. Tai, C. Wang, J. Li, and F. Huang, "Real-world super-resolution via kernel estimation and noise injection," in *CVPRW*, 2020.
- [57] X. Xu, Y. Ma, and W. Sun, "Towards real scene super-resolution with raw images," in *CVPR*, 2019.
- [58] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv*, 2015.
- [59] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.
- [60] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image translation with conditional adversarial networks," in *CVPR*, 2017.
- [61] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv*, 2016.
- [62] Z. Hu and M.-H. Yang, "Good regions to deblur," in *ECCV*, 2012.
- [63] A. Ignatov, N. Kobyshev, R. Timofte, K. Vanhoey, and L. Van Gool, "DSLR-quality photos on mobile devices with deep convolutional networks," in *ICCV*, 2017.
- [64] K. Yu, C. Dong, L. Lin, and C. Change Loy, "Crafting a toolchain for image restoration by deep reinforcement learning," in *CVPR*, 2018.
- [65] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
- [66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv*, 2014.



**Sanghyun Son** is a Ph.D. student in the Department of Electrical and Computer Engineering at the Seoul National University (SNU), Seoul, Korea. He graduated *summa cum laude* from Seoul National University with a B.S. degree in Electrical and Computer Engineering in 2017. He is interested in real-world computer vision problems including image restoration and enhancement, especially image super-resolution and its practical applications.



**Jaeha Kim** is a M.S. student in the Department of Electrical and Computer Engineering at the Seoul National University (SNU), Seoul, Korea. He graduated *summa cum laude* from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, with a B.S. degree in Electrical Engineering in 2019. His research field is image enhancement tasks in computer vision, especially real-world image super-resolution.



**Wei-Sheng Lai** is a software engineer at Google, CA, USA. He received the B.S. and M.S. degree in Electrical Engineering from the National Taiwan University, Taipei, Taiwan, and his PhD degree in Electrical Engineering and Computer Science at the University of California Merced in 2019.



**Ming-Hsuan Yang** is affiliated with Google, UC Merced, and Yonsei University. Yang serves as a program co-chair of IEEE International Conference on Computer Vision (ICCV) in 2019, program co-chair of Asian Conference on Computer Vision (ACCV) in 2014, and general co-chair of ACCV 2016. Yang served as an associate editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence, and is an associate editor of the International Journal of Computer Vision, Image and Vision Computing and Journal of Artificial Intelligence Research. He received the NSF CAREER award and Google Faculty Award. He is a Fellow of the IEEE.



**Kyoung Mu Lee** received the BS and MS degrees in control and instrumentation engineering from Seoul National University (SNU), Seoul, Korea, in 1984 and 1986, respectively, and the PhD degree in electrical engineering from the University of Southern California, in 1993. He is currently with the Department of ECE, Seoul National University as a professor. He has received several awards, in particular, the medal of merit and the Scientist of Engineers of the month award from the Korean Government in 2020 and 2018, respectively, the Most Influential Paper over the Decade Award by the IAPR Machine Vision Application in 2009, the ACCV Honorable Mention Award in 2007, the Okawa Foundation Research Grant Award in 2006, the Distinguished Professor Award from the college of Engineering of SNU in 2009, and both the Outstanding Research Award and the Shinyang Engineering Academy Award from the College of Engineering of SNU in 2010. He has served as an Associate Editor in Chief (AEIC) and an Associate Editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), an Associate Editor of the Machine Vision Application (MVA) Journal and the IPSJ Transactions on Computer Vision and Applications (CVA), the IEEE Signal Processing Letters (SPL), and an Area Editor of the Computer Vision and Image Understanding (CVIU). He also has served as a general chair of ICCV2019, ACMMM2018, and ACCV2018, a program chair of ACCV2012, a track chair of ICPR2020 and ICPR2012, and an area chair of CVPR, ICCV and ECCV many times. He was a distinguished lecturer of the Asia-Pacific Signal and Information Processing Association (APSIPA) for 2012-2013. He is an Advisory Board Member of the Computer Vision Foundation (CVF). More information can be found on his homepage <http://cv.snu.ac.kr/kmllee>.

## S1 DETAILS ABOUT THE LOW-PASS FILTERS

**Formulation of the low-pass filters.** We describe specific implementations of low-pass filters in the proposed LFL formulation. Our LFL utilizes conventional low-pass filters to extract low-frequency components from given images. Therefore, we replace the term  $\text{LPF}_*$  in (4) to a kernel representation  $k_{\text{HR}}$  and  $k_{\text{Down}}$  to simplify the description as follows:

$$\mathcal{L}_{\text{LFL}} = \left\| (\mathbf{I}_{\text{HR}} * k_{\text{HR}})_{\downarrow ms} - (\mathbf{I}_{\text{Down}} * k_{\text{Down}})_{\downarrow m} \right\|_1, \quad (\text{S1})$$

where  $m$  is a subsampling factor of the LR image, and (S1) is equivalent to (4) in our main manuscript. For example, our default  $\text{LFL}_m$  formulation corresponds to a 2D kernel  $k_{\text{Down}}$  of  $m \times m$  where  $k_{\text{Down}}(x, y) \equiv 1/m^2$ . We note that  $m = 16$  is used throughout our studies. In  $\times 2$  downsampling case, a kernel  $k_{\text{HR}}$  for HR images can be expressed as a box filter of  $32 \times 32$  with  $k_{\text{HR}}(x, y) \equiv 1/32^2$ . Here,  $(x, y)$  describes a coordinate system of the downsampling kernel, including a sub-pixel shift. Specifically, a center of the  $16 \times 16$  kernel, which is not a pixel, corresponds to  $k(0, 0)$  and neighboring 4 pixels are represented as  $k(\pm 0.5, \pm 0.5)$ , respectively. This formulation is useful for Gaussian kernels on an even-sized grid as follows:

$$k(x, y) = \frac{1}{Z} \exp\left(-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}\right), \quad (\text{S2})$$

where  $Z$  is a normalization factor so that  $\sum_{x,y} k(x, y) \equiv 1$ . For the proposed LFL, only isotropic cases are tested where  $\sigma_x$  and  $\sigma_y$  are equal. In the Gaussian cases, we follow a convention and set the kernel grid size to  $p \times p$  where  $p$  is the nearest power of two from  $6\sigma_x$ . While we adopt the filtering-based method for our LFL for simplicity, more complex formulations such as Wavelet can be introduced without losing generality.

**Selection of the low-pass filters.** An appropriate selection of the low-pass filter in our LFL plays an essential role. Therefore, we conduct an extensive ablation study to determine the low-pass filter when training the downsampler  $\mathcal{D}$ . Table S2 shows how different types and shapes of low-pass filters for the downsampler affect the SR results. We present the performance evaluation on the synthetic DIV2K dataset with a challenging anisotropic Gaussian kernel  $k_4$ . As shown in Table S2(a), a small box filter, e.g.,  $m = 2$ , may bias the training objective and degrade the following SR performance. On the other hand, a large box filter with  $m = 64$  operates as an extremely loose constraint and cannot contribute to preserving image contents across different scales. In Table S2(b), we have also introduced 2D Gaussian filters for the LFL. However, simple box filters have demonstrated relatively better performance. Thus, we use  $16 \times 16$  box filters for the low-pass filter  $\text{LPF}_m$  and  $32 \times 32$  for  $\text{LPF}_{ms}$  by default throughout our experiments.

## S2 DETAILS ABOUT THE SYNTHETIC KERNELS

We present a formulation of the  $\times 2$  synthetic downsampling kernels used for various experiments in our main manuscript. As we describe in Section 4.1,  $k_0$  denotes a widely-used MATLAB bicubic kernel. The other kernels,

TABLE S1

### Specifications of low-pass filters we use.

For box and Gaussian filters, weights are normalized so that their values are summed to 1. We note that a subsampling by  $ms$  and  $m$  follow after  $\text{LPF}_{ms}$  and  $\text{LPF}_s$ , respectively, to reduce image resolutions. More details about the filters are described in Appendix A.

Type	Filter	Size	Shape
2D Box	$\text{LPF}_{ms}$	$ms \times ms$	$ms \times ms$ box
	$\text{LPF}_s$	$m \times m$	$m \times m$ box
2D Gaussian	$\text{LPF}_{ms}$	$ms \times ms$	$\sigma_x = \sigma_y = s\sigma$
	$\text{LPF}_s$	$m \times m$	$\sigma_x = \sigma_y = \sigma$

TABLE S2

### Ablation study on the shapes and sizes of $\text{LPF}_m$ .

We train the LFL-based downsampler and the following SR model on DIV2K  $\times 2$  ( $k_4$ ) to observe how different low-pass filters affect the performance of our approach.

(a) Box filters for  $\text{LPF}_m$ 

Method \ Box size $m$	PSNR $\uparrow$ (dB) for $\times 2$ SR ( $k_4$ )					
	2	4	8	16	32	64
LFL + EDSR ( <b>Proposed</b> )	29.51	30.17	31.06	<u>31.57</u>	<b>31.58</b>	28.11

(b) Gaussian filters for  $\text{LPF}_m$ 

Method \ Gaussian sigma $\sigma$	PSNR $\uparrow$ for $\times 2$ SR ( $k_4$ )					
	0.8	1.2	1.6	2.0	2.5	3.0
LFL + EDSR ( <b>Proposed</b> )	29.64	30.24	30.42	30.62	30.96	30.75

TABLE S3

### Detailed parameters to implement the synthetic Gaussian kernels.

Fig. 5 in our main manuscript also visualizes each downsampling kernel in detail.

Kernel $k_i$	$\sigma_x$	$\sigma_y$	$\theta$	Type
$k_1$	1.0	1.0	$0^\circ$	Isotropic
$k_2$	1.6	1.6	$0^\circ$	Isotropic
$k_3$	1.0	2.0	$0^\circ$	Anisotropic
$k_4$	1.0	2.0	$29^\circ$	Anisotropic

i.e.,  $k_1 \sim k_4$ , are  $20 \times 20$  and sampled from a standard 2D Gaussian distribution following (S2). Table S3 describes the actual parameters used to instantiate our synthetic kernels. To validate the generalization ability of the proposed method, we do not resort to radial kernels that are relatively easy to model and introduce anisotropic kernels  $k_3$  and  $k_4$ . The most challenging case  $k_4$  further includes rotation of random degrees  $\theta$  and have neither vertical nor horizontal symmetries. For a larger  $\times 4$  downsampling factor, we follow an approach from KernelGAN [19] and convolve the same kernel twice to generate a larger one.

## S3 STABILITY OF THE PROPOSED METHODS

Since our LFL and ADL rely on unsupervised adversarial training, stability and reproducibility of the proposed method can be an essential issue. Therefore, we conduct five independent experiments for each of the five downsampling kernels  $k_0 \sim k_4$  at a scale factor of  $\times 2$  to analyze the stability of our training scheme. Fig. S1 shows the average performance of the following SR model after we train the downsampler using LFL and ADL, across five different kernel configurations on the synthetic DIV2K dataset. Even in the unsupervised learning framework, the SR model with our LFL and ADL schemes performs consistently with a

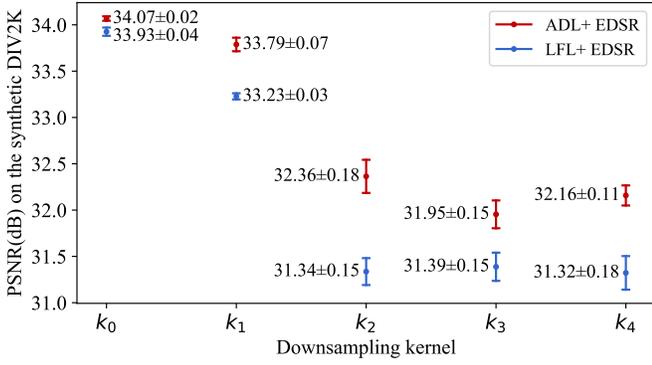


Fig. S1. **Stability analysis of the proposed methods.** We visualize the average performance of the baseline EDSR  $\times 2$  model and standard deviation from five runs on each downsampling kernel. Notably, ADL consistently outperforms LFL in all synthetic downsampling kernel configurations.

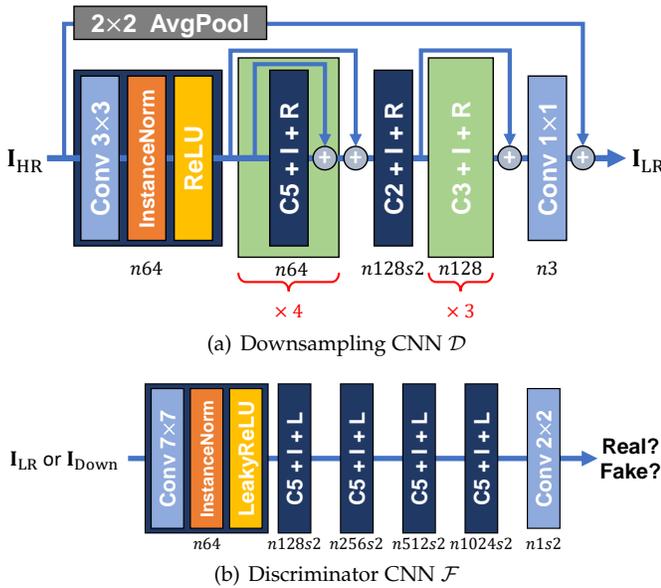


Fig. S2. **Our CNN architectures.**  $CK$  in the navy box denotes a  $K \times K$  convolutional layer, e.g., C5 for  $5 \times 5$ . For a sequence of the convolutional, instance normalization [61], and ReLU (or LeakyReLU [58]) activation layers, we use the term  $CK + I + R$  (or  $L$ ) for simplicity. The green block in (a) incorporates a shortcut connection wrapping around the  $CK + I + R$  sequence. We note that  $n$  refers to the number of output channels, and  $s$  describes the stride of the convolutional layer with a default value of 1, respectively.

small variation. Since the SR model performs stably across different experimental configurations, we report the result from one single run in the other sections.

## S4 DETAILED COMPARISON WITH KERNELGAN

Table 4 in our main manuscript shows that the proposed ADL + EDSR outperforms the KernelGAN + ZSSR combination by a significant margin even when only one LR image is available for the training. In this section, we use multiple images to train KernelGAN to demonstrate the advantage of our method when large-scale unpaired images are available. Table S4 shows extensive experimental results regarding different training datasets of the KernelGAN. We note that ZSSR is used to reconstruct  $\mathbf{I}_{SR}$  following KernelGAN by default unless mentioned otherwise.

First, in Cases 1 and 2, we modify KernelGAN to use 100 validation LR images as a training dataset and predict a shared downsampling kernel rather than calculate it for each image. This configuration demonstrates how KernelGAN operates on large-scale data. Using the estimated kernel, ZSSR is applied to each image independently. Compared to the original KernelGAN + ZSSR configuration, i.e., Case 0, using more images for kernel estimation has demonstrated inconsistent performance variations on synthetic kernel experiments  $k_0 \sim k_4$  in Case 1. For the kernels  $k_2$  and  $k_3$ , using more data has brought noticeable performance improvements. However, with the kernels  $k_0$ ,  $k_1$ , and  $k_4$ , using a single image yields better results.

The capacity of ZSSR is relatively smaller than recent state-of-the-art methods, which may limit the performance of the KernelGAN + ZSSR combination. Specifically, the model has only 0.2M parameters and uses a single image for training. Therefore, we introduce a larger EDSR-baseline model as an SR backbone network with 400 training samples in Case 2. Similar to the proposed LFL and ADL experiments in our main manuscript, we synthesize 400 LR images from DIV2K '0001.png'  $\sim$  '0400.png' using the estimated kernel from the KernelGAN model. The following EDSR is then trained on the synthetic LR-HR pairs. However, the final SR performance decreases, while EDSR-baseline has a larger capacity than ZSSR. Unlike our ADL which brings additional performance gains with a larger SR backbone (see ADL + EDSR and ADL + RRDB in Table 4), such behavior demonstrates that better fitting to the kernel from KernelGAN does not guarantee higher SR performance.

In Cases 3 and 4, we adopt the same dataset configuration as the proposed LFL and ADL, i.e., 400 HR images with unpaired 400 LR samples, when training KernelGAN. Therefore, the only difference between our and KernelGAN algorithms is model architectures (nonlinear CNN vs. deep linear generator) and loss functions (adaptive downsampling loss vs. kernel constraints). We first estimate the shared kernel  $k$  by feeding  $\mathbf{I}_{HR}$  to the deep linear generator, and the discriminator is optimized to distinguish the output of the downsampling model and real LR images  $\mathbf{I}_{LR}$  synthesized by a ground-truth kernel  $k_i$ . In Case 3, ZSSR is applied to '0801.png'  $\sim$  '0900.png' independently using a single shared kernel. In Case 4, we train EDSR similar to Case 2. We note that the only difference between Cases 1, 2, and Cases 3, 4 is a training dataset for the generator, i.e., downsampling model in KernelGAN.

To summarize, our ADL + EDSR or ADL + RRDB formulation show consistently better performance compared to KernelGAN regardless of the number of training data and model capacity.

## S5 NETWORK ARCHITECTURE

Fig. S2 illustrates CNN architectures we use throughout our main manuscript. The downsampling network adopts the residual connections [65] and instance normalization [61] strategy for easier optimization. A global residual connection [26] with  $2 \times 2$  average pooling operation is also introduced to provide a stable starting point. We note that the  $2 \times 2$  average pooling in Figure S2(a) does not operate as a restrictive prior which instabilizes the adversarial training

TABLE S4  
**Ablation study on using more data for the KernelGAN method.**

HR images correspond to inputs of the  $\mathbf{I}_{LR}$  generator, i.e., the deep linear generator for KernelGAN experiments and our downsampling network for the ADL configuration, while LR images are *real* samples for the discriminator network. We note that Case 0 and ADL correspond to Table 4 in our main manuscript. EDSR denotes a baseline version that has 1.4M parameters. All evaluations are done using DIV2K ‘0801.png’~‘0900.png.’

Case	Dataset for the $\mathbf{I}_{LR}$ Generator		SR model	PSNR $^{\uparrow}$ (dB) for $\times 2$ SR				
	HR image(s)	LR image(s)		$k_0$	$k_1$	$k_2$	$k_3$	$k_4$
1	‘0801’~‘0900’		ZSSR	21.54	26.41	30.55	31.18	27.68
2			EDSR	16.87	18.55	29.95	31.08	23.28
3	‘0001’~‘0400’	‘0401’~‘0800’	ZSSR	20.91	26.83	29.97	28.46	27.99
4			EDSR	16.11	20.35	29.26	24.85	19.68
0	$\mathbf{I}_{LR}$	$\mathbf{I}_{LR}$	ZSSR	22.32	26.42	30.44	29.10	29.12
ADL	‘0001’~‘0400’	‘0401’~‘0800’	EDSR	<b>34.07</b>	<b>33.68</b>	<b>32.51</b>	<b>32.08</b>	<b>32.05</b>

objective, since it does not force the output  $\mathbf{I}_{Down}$  to be a specific function of the input  $\mathbf{I}_{HR}$ . Unlike the KernelGAN [19] model, our downsampling CNN incorporates nonlinear ReLU activations and thus can learn a more generalized function. Our discriminator network is fully convolutional [60] and returns a  $2 \times 2$  probability map from a  $64 \times 64$  input patch. Both of the downsampling and discriminator CNNs are initialized with weights of random Gaussian  $\mathcal{N}(0, 0.02^2)$ .

## S6 DETAILS ABOUT THE HYPERPARAMETERS

We present detailed hyperparameters and experimental configurations that are not described in our main manuscript. In the downsampling task, i.e., Algorithm 1 and (2) in our main manuscript, we set the learning rate of  $\eta$  of downsampling and discriminator CNNs  $\mathcal{D}$  and  $\mathcal{F}$  to  $5 \times 10^{-5}$ . For each iteration, we use 32 samples of patch size  $128 \times 128$  as an input batch and generate LR images of  $64 \times 64$ . To train SR models, we use a batch size of 16 with  $48 \times 48$  input images. The learning rate is set to  $10^{-4}$ , similar to conventional approaches for deep image super-resolution [6], [9], [10]. The only differences are that we reduce the learning rate by half for every 50 epochs and our baseline EDSR [6] model is trained for 200 epochs, not 300, as validation performance does not change after then. In all experiments, pixel values are normalized from  $[0, 255]$  to  $[-1, 1]$ . We adopt the ADAM [66] optimizer with  $(\beta_1, \beta_2) = (0.9, 0.999)$  and  $\epsilon = 10^{-8}$  for all learnable parameters. It takes about 15 hours to learn the proposed downsampler with a single RTX 2080 Ti GPU on the synthetic DIV2K dataset. The learning time reduces to about 4 hours on the RealSR-V3 dataset as it contains fewer HR and LR samples.

## S7 ADDITIONAL QUALITATIVE COMPARISONS

We present more qualitative SR results in Figure S3, S4, and S5. While the IKC [17] model also reconstructs clean and sharp results for some specific synthetic cases, e.g.,  $k_1$  and  $k_2$ , our combination of the ADL + RRDB [8] generalizes well with a *fixed* hyperparameter configuration, regardless of synthetic or realistic inputs. As described in our main manuscript, more qualitative results can be found from our project page: <https://cv.snu.ac.kr/research/ADL>.

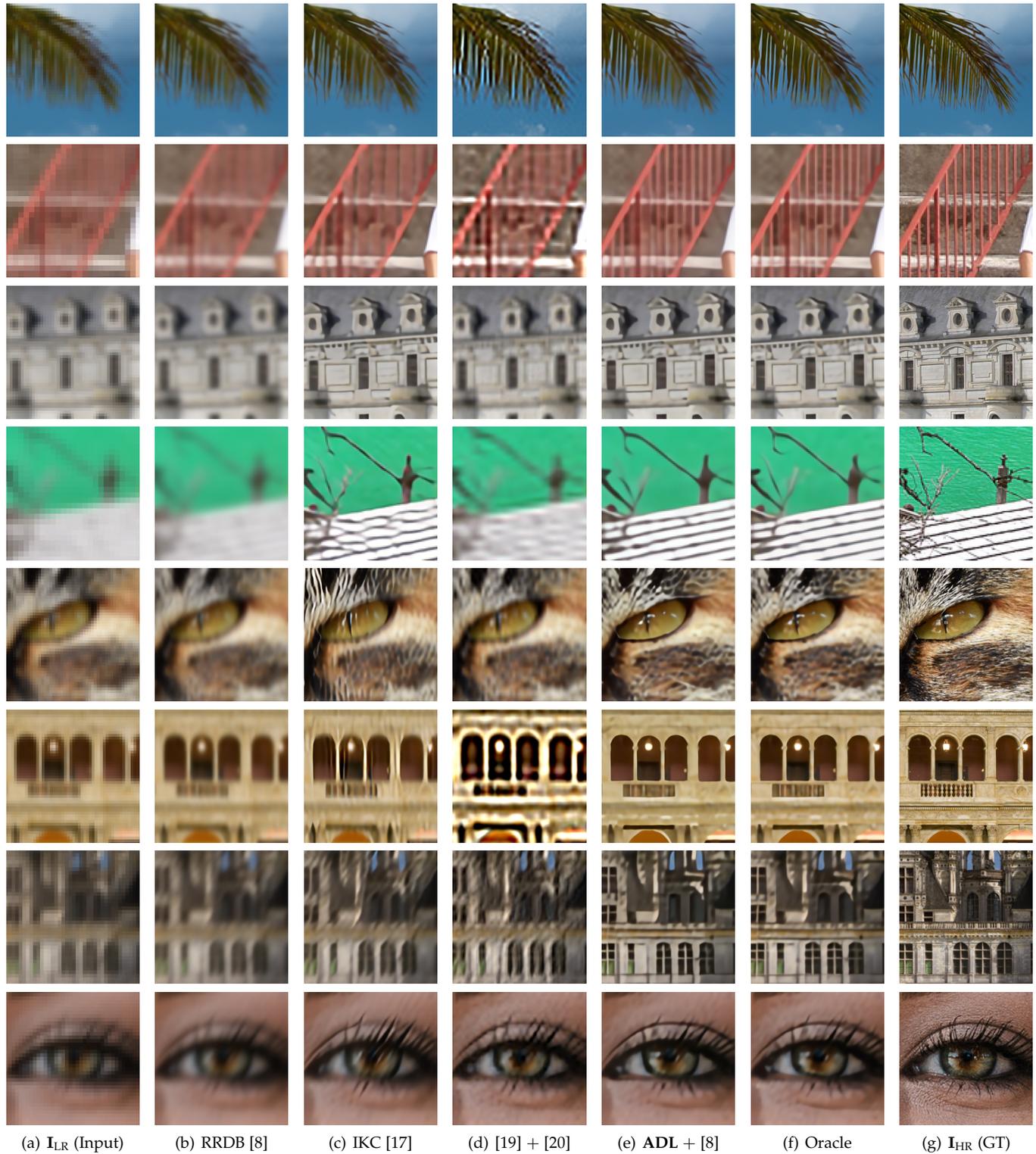


Fig. S3. **Additional qualitative  $\times 4$  SR results on the synthetic DIV2K [7] dataset.** From the top, patches are cropped from the DIV2K ‘0806.png ( $k_1$ ), ‘0825.png ( $k_1$ ), ‘0865.png ( $k_2$ ), ‘0807.png ( $k_2$ ), ‘0869.png ( $k_3$ ), ‘0884.png ( $k_3$ ), ‘0830.png ( $k_4$ ), and ‘0855.png ( $k_4$ ),’ respectively, where LR images are synthesized using corresponding downsampling kernels in the parenthesis ( $\cdot$ ).



Fig. S4. **Additional qualitative  $\times 4$  SR results on the RealSR-V3 [14] dataset.** From the top, patches are cropped from 'Canon/001.png', 'Canon/003.png', 'Canon/022.png', 'Canon/033.png', 'Nikon/004.png', 'Nikon/041.png', 'Nikon/049.png', and 'Nikon/050.png', respectively. Our approach (ADL + RRDB [8]) produces the least upsampling noise and artifacts in output images.

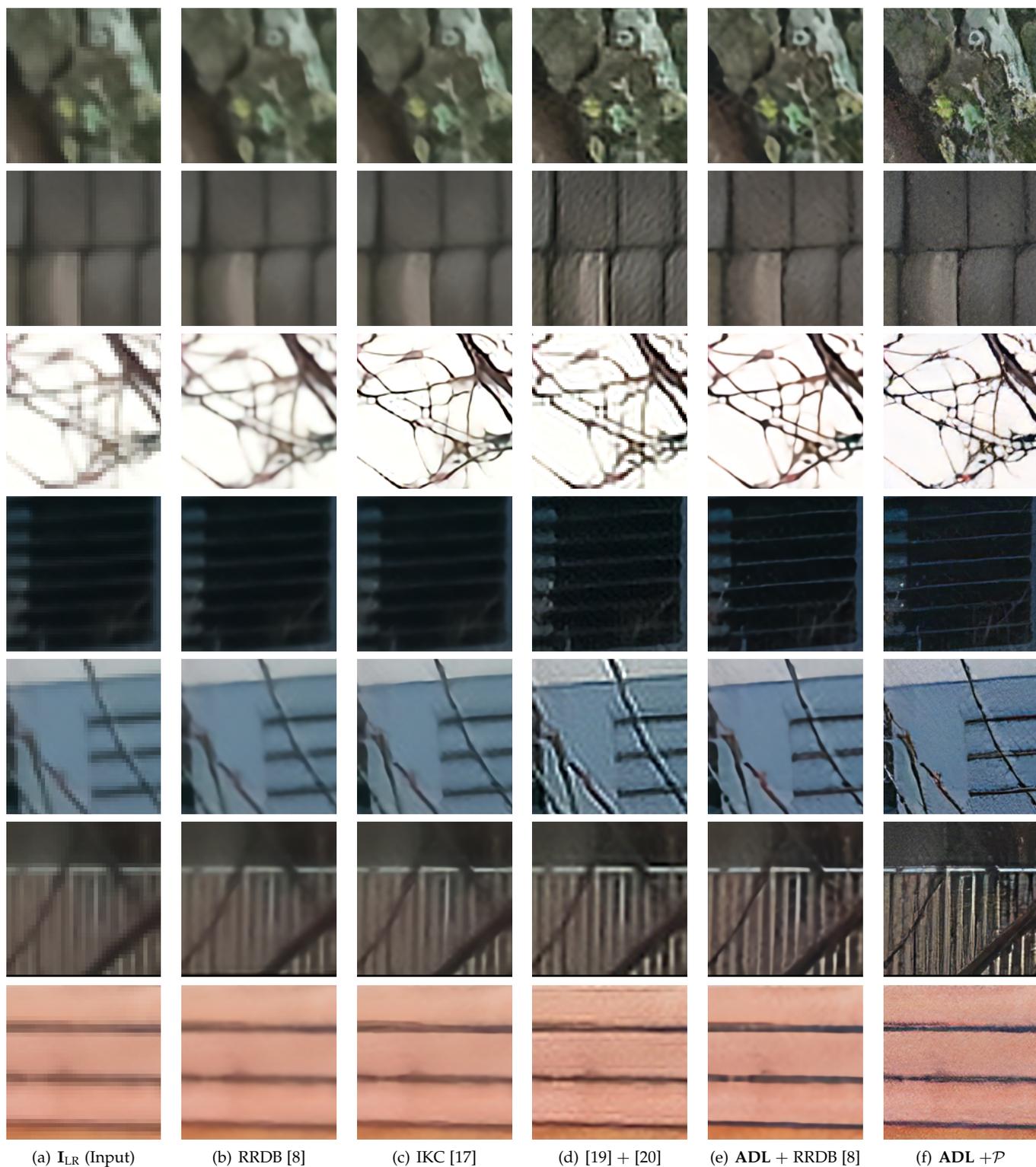


Fig. S5. **Additional Qualitative  $\times 4$  SR results on the DPED [63] dataset.** From the top, patches are cropped from the 'DPED-val' '14.png', '36.png', '45.png', '58.png (1)', '58.png (2)', '84.png', and '96.png' respectively. We note that  $\mathcal{P}$  refers to the perceptual RRDB model trained with (9) in our main manuscript. Compared to the other methods, our approaches (ADL + RRDB and ADL +  $\mathcal{P}$ ) reconstruct sharper edges and detailed structures from given real-world LR images.