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,

APPENDIX A
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 LPF in (4) to a kernel representation \( k_{HR} \) and \( k_{\text{Down}} \) to simplify the description as follows:

\[
L_{\text{LFL}} = \| (I_{\text{HR}} \ast k_{HR})_{\downarrow m,s} - (I_{\text{Down}} \ast k_{\text{Down}})_{\downarrow m} \|_1, \tag{A}
\]

where \( m \) is a subsampling factor of the LR image, and (A) is equivalent to (4) in our main manuscript. For example, our default LFL formulation corresponds to a 2D kernel \( k_{\text{Down}} \) of \( m \times m \) where \( k_{\text{Down}}(x, y) \equiv \frac{1}{m^2} \). We note that \( m = 16 \) is used throughout our studies. In \( \times 2 \) downsampling case, a kernel \( k_{HR} \) for HR images can be expressed as a box filter of \( 32 \times 32 \) with \( k_{HR}(x, y) \equiv \frac{1}{32^2} \). 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 \( (0, 0) \) and neighboring 4 pixels are represented as \( k(\pm 0.5, 0) \) and \( 0 \). This formulation is useful for Gaussian kernels on an even-sized grid as follows:

\[
k(x, y) = \frac{1}{Z} \exp \left( -\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2} \right), \tag{B}
\]

where \( Z \) is a normalization factor so that \( \sum_{x,y} k(x, y) = 1 \). 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 \( 6\sigma \). 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.

\* 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, hjykim2, kyoungmu@snu.ac.kr

W.-S. Lai and Ming-Hsuan are with Google Research. 
E-mail: lwsdai, minghsuan@google.com

<table>
<thead>
<tr>
<th>Type</th>
<th>Filter</th>
<th>Size</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D Box</td>
<td>LPF_{ms}</td>
<td>m \times ms</td>
<td>m \times ms box</td>
</tr>
<tr>
<td>2D Gaussian</td>
<td>LPF_{ms}</td>
<td>m \times m</td>
<td>σ_x = σ_y = sσ</td>
</tr>
<tr>
<td>2D Gaussian</td>
<td>LPF_s</td>
<td>m \times m</td>
<td>σ_x = σ_y = σ</td>
</tr>
</tbody>
</table>

TABLE A 
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 LPF_{ms} and LPF_s, respectively, to reduce image resolutions. More details about the filters are described in Appendix A.

<table>
<thead>
<tr>
<th>Method \ Box size m</th>
<th>PSNR^1(dB) for ( \times 2 ) SR (k_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFL + EDSR (Proposed)</td>
<td>29.51 30.17 31.06 31.57 31.58 28.11</td>
</tr>
</tbody>
</table>

TABLE B 
Ablation study on the shapes and sizes of LPF_m.
We train the LFL-based downsampler and the following SR model on DIV2K \( \times 2 \) to observe how different low-pass filters affect the performance of our approach.

(a) Box filters for LPF_m
<table>
<thead>
<tr>
<th>Method \ Box size m</th>
<th>PSNR^1(dB) for ( \times 2 ) SR (k_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFL + EDSR (Proposed)</td>
<td>29.64 30.24 30.42 30.62 30.96 30.75</td>
</tr>
</tbody>
</table>

(b) Gaussian filters for LPF_m
<table>
<thead>
<tr>
<th>Method \ Gaussian sigma σ</th>
<th>PSNR^1(dB) for ( \times 2 ) SR (k_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFL + EDSR (Proposed)</td>
<td>29.64 30.24 30.42 30.62 30.96 30.75</td>
</tr>
</tbody>
</table>

Therefore, we conduct an extensive ablation study to determine the low-pass filter when training the downsampler. Table B 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 2(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 2(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 LPF_m and \( 32 \times 32 \) for LPF_{ms} by default throughout our experiments.
APPENDIX B
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, i.e., $k_1 \sim k_4$, are $20 \times 20$ and sampled from a standard 2D Gaussian distribution following (B). Table C 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 [2] and convolve the same kernel twice to generate a larger one.

### Table C
<table>
<thead>
<tr>
<th>Kernel $k_i$</th>
<th>$\sigma_x$</th>
<th>$\sigma_y$</th>
<th>$\theta$</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>1.0</td>
<td>1.0</td>
<td>0$^\circ$</td>
<td>Isotropic</td>
</tr>
<tr>
<td>$k_2$</td>
<td>1.6</td>
<td>1.6</td>
<td>0$^\circ$</td>
<td>Isotropic</td>
</tr>
<tr>
<td>$k_3$</td>
<td>1.0</td>
<td>2.0</td>
<td>0$^\circ$</td>
<td>Anisotropic</td>
</tr>
<tr>
<td>$k_4$</td>
<td>1.0</td>
<td>2.0</td>
<td>29$^\circ$</td>
<td>Anisotropic</td>
</tr>
</tbody>
</table>

APPENDIX C
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 $\times 2$ and analyze the stability of our training scheme. Fig. A 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 small variation. Since the SR model performs stably across different experimental configurations, we report the result from one single run in the other sections.

APPENDIX D
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 D shows extensive experimental results regarding different training datasets of the KernelGAN. We note that ZSSR is used to reconstruct $I_{\text{SR}}$ 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’∼’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 \( I_{HR} \) to the deep linear generator, and the discriminator is optimized to distinguish the output of the downsampling model and real LR images \( I_{LR} \) synthesized by a ground-truth kernel \( k_1 \). In Case 3, ZSSR is applied to ’0801.png’∼’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.

APPENDIX E
Network Architecture

Fig. B illustrates CNN architectures we use throughout our main manuscript. The downsampling network adopts the residual connections [6] and instance normalization [14] strategy for easier optimization. A global residual connection [9] 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 B(a) does not operate as a restrictive prior which instabilizes the adversarial training objective, since it does not force the output \( I_{Down} \) to be a specific function of the input \( I_{HR} \). Unlike the KernelGAN [2] model, our downsampling CNN incorporates nonlinear ReLU activations and thus can learn a more generalized function. Our discriminator network is fully convolutional [8] 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 \( \mathcal{N}(0, 0.02^2) \).

APPENDIX F
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 \( D \) and \( F \) 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 [5], [11], [16]. The only differences are that we reduce the learning rate by half for every 50 epochs and our baseline EDSR [11] 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 [10] 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 down sampler 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.

APPENDIX G
Additional Qualitative Comparisons

We present more qualitative SR results in Figure C, D, and E. While the IKC [4] model also reconstructs clean and sharp results for some specific synthetic cases, e.g., \( k_1 \) and \( k_2 \), our combination of the AKL + RRDB [15] 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.

### TABLE D
Ablation study on using more data for the KernelGAN method.

<table>
<thead>
<tr>
<th>Case</th>
<th>Dataset for the ( I_{LR} ) Generator SR model</th>
<th>( k_0 )</th>
<th>( k_1 )</th>
<th>( k_2 )</th>
<th>( k_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0801’~0900’ ZSSR</td>
<td>21.54</td>
<td>26.41</td>
<td>30.55</td>
<td>31.18</td>
</tr>
<tr>
<td>2</td>
<td>0401’ ZSSR</td>
<td>16.87</td>
<td>18.55</td>
<td>29.95</td>
<td>31.08</td>
</tr>
<tr>
<td>3</td>
<td>0001’~0400’ ZSSR</td>
<td>20.91</td>
<td>26.83</td>
<td>29.97</td>
<td>28.46</td>
</tr>
<tr>
<td>4</td>
<td>0401’~0800’ ZSSR</td>
<td>16.11</td>
<td>20.35</td>
<td>29.26</td>
<td>24.85</td>
</tr>
<tr>
<td>0</td>
<td>( I_{LR} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADL</td>
<td>0801’~0400’ ( I_{LR} )</td>
<td>22.32</td>
<td>26.42</td>
<td>30.44</td>
<td>29.10</td>
</tr>
<tr>
<td></td>
<td>ZSSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EDSR</td>
<td>34.07</td>
<td>33.68</td>
<td>32.51</td>
<td>32.08</td>
</tr>
</tbody>
</table>

HR images correspond to inputs of the \( I_{LR} \) 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’.
REFERENCES

Fig. C. Additional qualitative ×4 SR results on the synthetic DIV2K [1] dataset. From the top, patches are cropped from the DIV2K ‘0806.png (k₁),’ ‘0825.png (k₁),’ ‘0865.png (k₂),’ ‘0807.png (k₂),’ ‘0869.png (k₃),’ ‘0884.png (k₃),’ ‘0830.png (k₄),’ and ‘0855.png (k₄),’ respectively, where LR images are synthesized using corresponding downsampling kernels in the parenthesis (·).
Fig. D. Additional qualitative ×4 SR results on the RealSR-V3 [3] 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 [15]) produces the least upsampling noise and artifacts in output images.
Fig. E. **Additional Qualitative \times 4 SR results on the DPED [7] 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 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 + P) reconstruct sharper edges and detailed structures from given real-world LR images.