
Shady Abu Hussein
Tel Aviv University, Israel
shaday@mail.tau.ac.il

Tom Tirer
Tel Aviv University, Israel
tomtirer@mail.tau.ac.il

Raja Giryes
Tel Aviv University, Israel
raja@tauex.tau.ac.il

Abstract

The single image super-resolution task is one of the most examined inverse problems in the past decade. In the recent years, Deep Neural Networks (DNNs) have shown superior performance over alternative methods when the acquisition process uses a fixed known downscaling kernel—typically a bicubic kernel. However, several recent works have shown that in practical scenarios, where the test data mismatch the training data (e.g. when the downscaling kernel is not the bicubic kernel or is not available at training), the leading DNN methods suffer from a huge performance drop. Inspired by the literature on generalized sampling, in this work we propose a method for improving the performance of DNNs that have been trained with a fixed kernel on observations acquired by other kernels. For a known kernel, we design a closed-form correction filter that modifies the low-resolution image to match one which is obtained by another kernel (e.g. bicubic), and thus improves the results of existing pre-trained DNNs. For an unknown kernel, we extend this idea and propose an algorithm for blind estimation of the required correction filter. We show that our approach outperforms other super-resolution methods, which are designed for general downscaling kernels.

1. Introduction

The task of Single Image Super-Resolution (SISR) is one of the most examined inverse problems in the past decade [11, 13, 37, 8]. In this problem, the goal is to reconstruct a latent high-resolution (HR) image from its low-resolution (LR) version obtained by an acquisition process that includes low-pass filtering and sub-sampling. In the recent years, along with the developments in deep learning, many SISR methods that are based on Deep Neural Networks (DNNs) have been proposed [7, 17, 21, 20, 41, 35, 15].

Typically, the performance of SISR approaches is evaluated on test sets with a fixed known acquisition process, e.g. a bicubic downscaling kernel. This evaluation methodology allows to prepare large training data, which are based on ground truth HR images and their LR counterparts synthetically obtained through the known observation model. DNNs, which have been exhaustively trained on such training data, clearly outperform other alternative algorithms, e.g. methods that are based on hand-crafted prior models such as sparsity or non-local similarity [8, 13, 37].

Recently, several works have shown that in practical scenarios where the test data mismatch the training data, the leading DNN methods suffer from a huge performance drop [38, 27, 31]. Such scenarios include a downscaling kernel which is not the bicubic kernel and is not available at the training phase. A primary example is an unknown kernel that needs to be estimated from the LR image at test time.

Several recent SISR approaches have proposed different strategies for enjoying the advantages of deep learning while mitigating the restriction of DNNs to the fixed kernel assumption made in the training phase. These strategies include: modifying the training phase such that it covers a predefined set of downscaling kernels [39, 14]; using DNNs to capture only a natural-image prior which is decoupled from the SISR task [38, 5]; or completely avoid any offline training and instead train a CNN super-resolver from scratch at test time [32, 27].

Contribution. In this work we take a different strategy, inspired by the generalized sampling literature [9, 33], for handling LR images obtained by arbitrary downscaling kernels. Instead of completely ignoring the state-of-the-art DNNs that have been trained for the bicubic model, as done by other prior works, we propose a method that transforms the LR image to match one which is obtained by the bicubic kernel. The modified LR can then be inserted into existing leading super-resolvers, such as DBPN [15], RCAN [41], and proSR [35], thus, improving their performance significantly on kernels they have not been trained on. The proposed transformation is performed using a correction filter, which has a closed-form expression when the true (non-bicubic) kernel is given.

In the “blind” setting, where the kernel is unknown, we extend our approach and propose an algorithm that esti-
mates the required correction. The proposed approach outperforms other super-resolution methods in various practical scenarios. See example results in Figure 1 for the non-blind setting.

2. Related Work

In the past five years many works have employed DNNs for the SISR task, showing a great advance in performance with respect to the reconstruction error [7, 15, 17, 21, 35, 41] and the perceptual quality [4, 20, 25, 34]. However, one main disadvantage of DNN-based SISR methods is their sensitivity to the LR image formation model. A network performance tends to drop significantly if it has been trained for one acquisition model and then been tested on another [27, 31, 38].

Recently, different SISR strategies have been proposed with the goal of enjoying the advantages of deep learning while mitigating the restriction of DNNs to the fixed kernel assumption made in the training phase. One approach trains a CNN super-resolver that gets as inputs both the LR image and the degradation model, and assumes that the downsampling kernels belong to a certain set of Gaussian filters [14, 39]. Another approach builds on the structural prior of CNNs, which promotes signals with spatially recurring patterns (e.g. natural images) and thus allows to train a super-resolver CNN from scratch at test time [27, 32]. Another line of work recovers the latent HR image by minimizing a cost function that is composed of a fidelity term (e.g. least squares or back-projection [30]) and a prior term, where
Figure 2: Downsampling, correction, and upsampling operators for single image super-resolution: (a) Downsampling operator composed of convolution with a kernel \( k \) and sub-sampling by factor of \( \alpha \); (b) Correction operator composed of convolution with a correction filter \( h \); (c) Upsampling operator composed of up-sampling by factor of \( \alpha \) and convolution with a (flipped) kernel \( k_{bicub} \). In our approach, \( H \) is computed for \( S^* \) and \( R \), but then we replace \( R \) with a pre-trained DNN super-resolver.

3. The Proposed Method

The single image super resolution (SISR) acquisition model, can be formulated as

\[
y = (x \ast k) \downarrow \alpha, \tag{1}
\]

where \( x \in \mathbb{R}^n \) represents the latent HR image, \( y \in \mathbb{R}^m \) represents the observed LR image, \( k \in \mathbb{R}^d \) is the (anti-aliasing) blur kernel, \( \downarrow \alpha \) denotes the linear convolution operator, and \( \downarrow \alpha \) denotes sub-sampling operator with stride of \( \alpha \). Under the common fashion of dropping the edges of \( x \ast k \), such that it is in \( \mathbb{R}^n \), we have that \( m = \lceil n/\alpha \rceil \).

Note that Equation (1) can be written in a more elegant way as

\[
y = S^*x, \tag{2}
\]

where \( S^* : \mathbb{R}^n \to \mathbb{R}^m \) is a linear operator that encapsulates the entire downsampling operator, i.e., \( S^* \) is a composition of blurring followed by sub-sampling. The downsampling operator \( S^* \) is presented in Figure 2(a).
Most SISR deep learning methods, e.g. [7, 17, 21, 20, 41, 35, 15], assume that the observations are obtained using the bicubic kernel. Let us denote by $\mathcal{R}^*$ the associated downsampling operator (essentially, $\mathcal{R}^*$ coincides with the previously defined $S^*$ if $k$ is the bicubic kernel $k_{bicub}$). The core idea of our approach is to modify the observations $y = S^*x$, obtained for an arbitrary downscaling kernel $k$, such that they mimic the (unknown) "ideal observations" $y_{bicub} = \mathcal{R}^*x$, which can be fed into pre-trained DNN models.

In what follows, we present a method to (approximately) achieve this goal using the correction filter tool, adopted from the generalized sampling literature. First, we consider the non-blind setting, where the downscaling kernel is known, and thus $S^*$ is known. In this case, we obtain a closed-form expression for the required correction filter, which depends on $k$ (and on $k_{bicub}$). Later, we extend the approach to the blind setting, where $k$ is unknown. In this case, we propose a technique for estimating the correction filter from the LR image $y$.

### 3.1. The non-blind setting

In the non-blind setting, both the downscaling kernel $k$ and the target kernel $k_{bicub}$ are known. Therefore, the downsampling operators $S^*$ and $\mathcal{R}^*$ are known as well. Using common notations from generalized sampling literature [9], let us denote by $S$ and $\mathcal{R}$ the adjoint operators of $S^*$ and $\mathcal{R}^*$, respectively. The operator $\mathcal{R} : \mathbb{R}^m \to \mathbb{R}^n$ is an upsampling operator that restores a signal in $\mathbb{R}^n$ from $m$ samples, associated with the downsampling operator $\mathcal{R}^*$. In the context of our work, when $\mathcal{R}$ is applied on a vector it pads it with $n - m$ zeros ($\alpha - 1$ zeros between each two entries) and convolves it with a flipped version of $k_{bicub}$. The upsampling operator $\mathcal{R}$ is presented in Figure 2(c). A similar definition holds for $S : \mathbb{R}^m \to \mathbb{R}^n$ with the kernel $k$.

The key goal of generalized sampling theory is to identify signal models and sampling systems that allow for perfect recovery. Therefore, to proceed, let us make the following assumption.

**Assumption 1.** The signal $x$ can be perfectly recovered from its samples $\mathcal{R}^*x$ by the operator $(\mathcal{R}^*\mathcal{R})^{-1}$, i.e.

$$x = \mathcal{R}(\mathcal{R}^*\mathcal{R})^{-1}\mathcal{R}^*x. \quad (3)$$

Assumption 1 essentially states that the latent image $x$ resides in the linear subspace spanned by the bicubic kernel. Therefore, it can be perfectly recovered from the observations $y_{bicub} = \mathcal{R}^*x$ by applying the pseudoinverse of $\mathcal{R}^*$ on $y_{bicub}$, i.e. by the estimator $\hat{x} = \mathcal{R}(\mathcal{R}^*\mathcal{R})^{-1}y_{bicub}$. Even though Assumption 1 does not hold for natural images, it is motivated by the fact that there are many DNN methods that can handle observations of the form $\mathcal{R}^*x$ quite well.

However, since we are given observations that are obtained by a different downsampling kernel, $y = S^*x$, let us propose a different estimator $\hat{x} = \mathcal{R}^*y$, where $\mathcal{H} : \mathbb{R}^m \to \mathbb{R}^n$ is a correction operator. This recovery procedure is presented in Figures 2(b)+2(c). The following theorem presents a condition and a formula for $\mathcal{H}$ under which perfect recovery is possible under Assumption 1.

**Theorem 2.** Let $y = S^*x$, $\hat{x} = \mathcal{R}^*y$, and assume that Assumption 1 holds. Then, if

$$\text{null}(S^*) \cap \text{range}(\mathcal{R}) = \{0\}, \quad (4)$$

we have that $\hat{x} = x$ for

$$\mathcal{H} = (S^*\mathcal{R})^{-1} : \mathbb{R}^m \to \mathbb{R}^m. \quad (5)$$

**Proof.** Note that

$$\hat{x} = \mathcal{R}^*y = \mathcal{R}^*S^*x = \mathcal{R}^*S^*(\mathcal{R}^*\mathcal{R})^{-1}\mathcal{R}^*x, \quad (6)$$

where the last equality follows from Assumption 1. Next, (4) implies that the operator $(S^*\mathcal{R})$ is invertible. Thus, setting $\mathcal{H}$ according to (5) is possible, and we get

$$\hat{x} = \mathcal{R}(S^*\mathcal{R})^{-1}S^*\mathcal{R}(\mathcal{R}^*\mathcal{R})^{-1}\mathcal{R}^*x = \mathcal{R}(\mathcal{R}^*\mathcal{R})^{-1}\mathcal{R}^*x = x, \quad (7)$$

where the last equality follows from Assumption 1.

Theorem 2 is presented in operator notations to simplify the derivation. In the context of SISR (i.e. with the previous definitions of $S^*$ and $\mathcal{R}$), the operator $\mathcal{H} = (S^*\mathcal{R})^{-1}$ can be applied simply as a convolution with a correction filter $h_0$, given by

$$h_0 = \text{IDFT} \left\{ \frac{1}{\text{DFT} \{ (k \ast \text{flip}(k_{bicub})) \downarrow_{\alpha} \}} \right\}, \quad (8)$$

where $\text{DFT} (\cdot)$ and $\text{IDFT} (\cdot)$ denote the Discrete Fourier Transform and its inverse respectively.\(^1\)

In practice, instead of using the weak estimator $\hat{x} = \mathcal{R}^*y$ that does not use any natural-image prior, we propose to recover the HR image by

$$\hat{x} = f(h \ast y), \quad (9)$$

where $f(\cdot)$ is a DNN super-resolver that has been trained under the assumption of bicubic kernel and $h$ is a modified

\(^1\)Using DFT allows fast implementation of cyclic convolutions. When it is used for linear convolutions, edge artifacts need to be ignored.
Table 1: Non-blind super-resolution comparison on Set14. Each cell displays PSNR [dB] (left) and SSIM (right).

<table>
<thead>
<tr>
<th>Scale</th>
<th>Gaussian std = 1.5/√2</th>
<th>Gaussian std = 2.5/√2</th>
<th>Box of width = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZSSR</td>
<td>2</td>
<td>28.107 / 0.829</td>
<td>27.954 / 0.806</td>
</tr>
<tr>
<td>SRMD</td>
<td>2</td>
<td>32.493 / 0.878</td>
<td>29.923 / 0.812</td>
</tr>
<tr>
<td>DBPN</td>
<td>2</td>
<td>30.078 / 0.850</td>
<td>26.366 / 0.734</td>
</tr>
<tr>
<td>ProSR</td>
<td>2</td>
<td>30.073 / 0.849</td>
<td>26.371 / 0.734</td>
</tr>
<tr>
<td>RCAN</td>
<td>2</td>
<td>30.118 / 0.851</td>
<td>26.389 / 0.736</td>
</tr>
<tr>
<td>RCAN + our correction</td>
<td>2</td>
<td>34.043 / 0.904</td>
<td>33.251 / 0.895</td>
</tr>
<tr>
<td>SRMD</td>
<td>4</td>
<td>25.642 / 0.701</td>
<td>25.361 / 0.683</td>
</tr>
<tr>
<td>DBPN</td>
<td>4</td>
<td>26.877 / 0.718</td>
<td>25.350 / 0.674</td>
</tr>
<tr>
<td>ProSR</td>
<td>4</td>
<td>25.067 / 0.685</td>
<td>23.890 / 0.645</td>
</tr>
<tr>
<td>ProSR + our correction</td>
<td>4</td>
<td>28.680 / 0.775</td>
<td>28.267 / 0.766</td>
</tr>
<tr>
<td>RCAN</td>
<td>4</td>
<td>25.033 / 0.683</td>
<td>23.882 / 0.645</td>
</tr>
<tr>
<td>RCAN + our correction</td>
<td>4</td>
<td>28.609 / 0.772</td>
<td>28.220 / 0.764</td>
</tr>
</tbody>
</table>

correction filter, given by

$$h = \text{IDFT} \left\{ \frac{\text{DFT} \{ (k_{\text{bicub}} \ast \text{flip}(k_{\text{bicub}})) \} \downarrow \alpha}{\text{DFT} \{ (k \ast \text{flip}(k_{\text{bicub}})) \} \downarrow \alpha} \right\} \right\}$$

$$\triangleq \text{IDFT} \left\{ \frac{F_{\text{numer}}}{F_{\text{denom}}} \right\}.$$ (10)

Let us explain the idea behind the estimator in (9). Since the inverse mapping $f(\cdot)$ assumes bicubic downsampling $(R^r \times R^c)$, it can be interpreted as incorporating $R(R^r \ast R)^{-1}$ with a (learned) prior. Therefore, unlike $h_0$ in (8) which is followed by $R$, here the correction filter should also compensate for the operation $(R^r \ast R)^{-1}$ which is implicitly done in $f(\cdot)$. This explains the term in the numerator of $h$ (compared to 1 in the numerator of $h_0$). To ensure numerical stability, we slightly modify (10), and compute $h$ using

$$h = \text{IDFT} \left\{ \frac{F_{\text{numer}}}{|F_{\text{denom}}|^2 + \epsilon} \right\},$$ (11)

where $\epsilon$ is a small regularization parameter. Regarding the choice of $f(\cdot)$, in our experiments we use DBPN [15], RCAN [41], and proSR [35], but in general any other method with state-of-the-art performance (for bicubic kernel) is expected to give good results.

Note that the theoretical motivation for our strategy requires that the condition in (4) holds. This condition can be inspected by comparing the bandwidth of the kernels $k$ and $k_{\text{bicub}}$ in the frequency domain. As $k$ is commonly a low-pass filter (and so is $k_{\text{bicub}}$), the condition requires that the passband of $k_{\text{bicub}}$ is contained in the passband of $k$. Yet, as shown in the experiments section, our approach yields a significant improvement even when the passband of $k$ is moderately smaller than the passband of $k_{\text{bicub}}$. Furthermore, we observe that even for very blurry LR images performance improvement can be obtained by increasing the regularization parameter in (11). We refer the reader to the supplementary material for more details.

### 3.2. The blind setting

In the blind setting, the downsampling kernel $k$ is unknown. Therefore, we cannot compute the correction filter $h$ using (11), and extending our approach to this setting requires to estimate $k$ and $h$ from the LR image $y$.

To this end, we propose to estimate $k$ as the minimizer of the following objective function

$$\xi(k) = ||y - S^* f(H(y))||_{\text{Hub}} + ||m_{\text{cen}} \cdot k||_1 + ||k||_1,$$ (12)

where $|| \cdot ||_{\text{Hub}}$ is Huber loss [16], the operator $H$ is filtering with $h$ given in (11), $S^*$ is the downsampling operator, $f(\cdot)$ is the given SR network, and $m_{\text{cen}}$ is given by

$$m_{\text{cen}}(x, y) = 1 - e^{-\frac{(x^2 + y^2)}{2\alpha^2}},$$

where $\alpha$ is the scale factor. Note that the two operators $H$ and $S^*$ depend on the kernel $k$. The last two terms in (12) are regularizers: the last term promotes sparsity of $k$ and the penultimate term centralizes its density.

Inspired by [2, 3], we choose to parameterize the latent $k$ by a linear CNN composed of 4 layers, i.e. $k = k_0 \ast k_1 \ast k_2 \ast k_3$, where $\{k_n\}_{n=0}^3$ are of size $33 \times 33$ and
Table 2: Non-blind super-resolution comparison on BSD100. Each cell displays PSNR [dB] (left) and SSIM (right).

<table>
<thead>
<tr>
<th>Method</th>
<th>Scale</th>
<th>Gaussian std = 1.5/√2</th>
<th>Gaussian std = 2.5/√2</th>
<th>Scale</th>
<th>Gaussian std = 3.5/√2</th>
<th>Gaussian std = 4.5/√2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZSSR</td>
<td>2</td>
<td>29.339 / 0.822</td>
<td>26.415 / 0.715</td>
<td>4</td>
<td>25.115 / 0.651</td>
<td>24.348 / 0.625</td>
</tr>
<tr>
<td>SRMD</td>
<td>2</td>
<td>26.591 / 0.803</td>
<td>29.294 / 0.838</td>
<td>4</td>
<td>25.735 / 0.704</td>
<td>26.432 / 0.707</td>
</tr>
<tr>
<td>DBPN</td>
<td>2</td>
<td>29.512 / 0.827</td>
<td>26.371 / 0.711</td>
<td>4</td>
<td>25.268 / 0.662</td>
<td>24.357 / 0.628</td>
</tr>
<tr>
<td>DBPN + correction</td>
<td>2</td>
<td>32.300 / 0.884</td>
<td>31.875 / 0.878</td>
<td>4</td>
<td>27.690 / 0.740</td>
<td>27.474 / 0.733</td>
</tr>
<tr>
<td>ProSR</td>
<td>2</td>
<td>29.513 / 0.827</td>
<td>26.381 / 0.711</td>
<td>4</td>
<td>25.237 / 0.661</td>
<td>24.353 / 0.628</td>
</tr>
<tr>
<td>ProSR + correction</td>
<td>2</td>
<td>32.276 / 0.884</td>
<td>31.899 / 0.878</td>
<td>4</td>
<td>27.645 / 0.738</td>
<td>27.455 / 0.733</td>
</tr>
<tr>
<td>RCAN</td>
<td>2</td>
<td>29.558 / 0.829</td>
<td>26.397 / 0.713</td>
<td>4</td>
<td>25.281 / 0.663</td>
<td>24.373 / 0.629</td>
</tr>
<tr>
<td>RCAN + correction</td>
<td>2</td>
<td>32.368 / 0.886</td>
<td>31.876 / 0.878</td>
<td>4</td>
<td>27.626 / 0.739</td>
<td>27.399 / 0.732</td>
</tr>
</tbody>
</table>

*Algorithm 1: Correction filter estimation*

**Input:** $y$, $k_{bicub}$, $\alpha$, $f(\cdot)$.  
**Output:** $\hat{h}$ an estimate for $h$.  
**Params:** $k^{(0)} = k_{bicub}$, $i = 0$, $\epsilon = 10^{-14}$, $\gamma = 10^{-4}$, $N_{iter} = 250$  
while $i < N_{iter}$ do  
  $i = i + 1$;  
  Compute $h^{(i)}$ using (11) (for $k^{(i-1)}$, $\alpha$ and $\epsilon$);  
  $x_h^{(i)} = f(h^{(i)} \ast y)$;  
  $\hat{y}^{(i)} = (x_h^{(i)} \ast k^{(i-1)}) \downarrow \alpha$;  
  $\xi(k^{(i-1)}) = \|y - \hat{y}^{(i)}\|_{Hub} + \|m_{cen} \cdot k^{(i-1)}\|_1 + \|k^{(i-1)}\|_1$;  
  $k^{(i)} = \text{Adam update (for } \xi(k^{(i-1)}) \text{ with LR } \gamma)$;  
end  
$h = h^{(i)}$.

$k_\alpha$ is of size $32 \times 32$. The minimization of (12) with respect to $k$ is performed by 250 iterations of Adam [18] with learning rate of $10^{-4}$, initialized with $k^{(0)} = k_{bicub}$. The proposed procedure is described in Algorithm 1. Note that at each iteration we obtain estimates for both the downscaling kernel $k$ and correction filter $h$. The final estimator of $h$ is then used in (9) to reconstruct the HR image, similarly to the non-blind setting.

4. Experiments

In this section we examine the performance and improvement due to our correction filter approach in the non-blind and blind settings, using three different off-the-shelf DNN super-resolvers that serve as $f(\cdot)$ in (9): DBPN [15], RCAN [41], and proSR [35]. We compare our approach to other methods that receive the downscaling kernel $k$ (or its estimation in the blind setting) as an input: ZSSR [27] and SRMD [39]. We also compare our method to DPSR [40], however, since its results are extremely inferior to the other strategies (e.g. about 10 dB lower PSNR) they are deferred to the supplementary material. All the experiments are performed with the official code of each method. We refer the reader to the supplementary material for more results.

4.1. The non-blind setting

In this section, we assume that the downscaling kernel $k$ is known. Therefore, the correction filter $h$ can be computed directly using (11). We examine scenarios with scale factors of 2 and 4. For scale factor of 2, we use Gaussian kernels with standard deviation $\sigma = 1.5/\sqrt{2}$ and $\sigma = 2.5/\sqrt{2}$, and box kernel of size $4 \times 4$. For scale factor of 4, we use Gaussian kernels with standard deviation $\sigma = 3.5/\sqrt{2}$ and $\sigma = 4.5/\sqrt{2}$, and box kernel of size $8 \times 8$.

The results are presented in Tables 1 and 2 for the test-
sets Set14 and BSD100, respectively. Figures 1 and 4 present several visual results. It can be seen that the proposed filter correction approach significantly improves the results of DBPN, RCAN, and proSR, which have been trained for the (incorrect) bicubic kernel. Moreover, note that the filter-corrected applications of DBPN, RCAN, and proSR, also outperform SRMD and ZSSR, while the plain applications of DBPN, RCAN, and proSR are inferior to SRMD.

As explained in Section 3, the proposed approach is based on mimicking the (unknown) “ideal” LR image $y_{bicub} = R^*x$ by the corrected LR image $\hat{y}$. The high PSNR and SSIM results between such pairs of images, which are presented in Table 3 for Set14, verify that this indeed happens. Figure 3 shows a visual comparison.

**Inference run-time.** Computing the correction filter requires a negligible amount of time, so it does not change the run-time of an off-the-shelf DNN. Using NVIDIA RTX 2080ti GPU, the per image run-time of all the methods except ZSSR is smaller than 1 second (because no training is done in the test phase), while ZSSR requires approximately 2 minutes per image.

### 4.2. The blind setting

In this section, we repeat previous experiments, but without the assumption that the downscaling kernel $k$ is known. Therefore, to apply our approach we first estimate the correction filter using Algorithm 1, and then use this estimation...
Table 4: Blind super-resolution comparison on Set14. Each cell displays PSNR [dB] (left) and SSIM (right).

<table>
<thead>
<tr>
<th>Scale</th>
<th>Gaussian std = 1.5/√2</th>
<th>Gaussian std = 2.5/√2</th>
<th>Box of width = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>KernelGAN</td>
<td>2</td>
<td>26.381 / 0.785</td>
<td>28.868 / 0.807</td>
</tr>
<tr>
<td>DBPN</td>
<td>2</td>
<td>30.078 / 0.85</td>
<td>26.366 / 0.734</td>
</tr>
<tr>
<td>DBPN + our estimated correction</td>
<td>2</td>
<td>28.461 / 0.842</td>
<td>28.037 / 0.794</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scale</th>
<th>Gaussian std = 3.5/√2</th>
<th>Gaussian std = 4.5/√2</th>
<th>Box of width = 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>KernelGAN</td>
<td>4</td>
<td>24.424 / 0.673</td>
<td>25.174 / 0.669</td>
</tr>
<tr>
<td>DBPN</td>
<td>4</td>
<td>25.067 / 0.685</td>
<td>23.890 / 0.645</td>
</tr>
<tr>
<td>DBPN + our estimated correction</td>
<td>4</td>
<td>28.184 / 0.764</td>
<td>25.542 / 0.699</td>
</tr>
</tbody>
</table>

Table 5: Blind super-resolution comparison on BSD100. Each cell displays PSNR [dB] (left) and SSIM (right).

<table>
<thead>
<tr>
<th>Scale</th>
<th>Gaussian std = 1.5/√2</th>
<th>Gaussian std = 2.5/√2</th>
<th>Gaussian std = 3.5/√2</th>
<th>Gaussian std = 4.5/√2</th>
</tr>
</thead>
<tbody>
<tr>
<td>KernelGAN</td>
<td>2</td>
<td>26.615 / 0.773</td>
<td>28.244 / 0.780</td>
<td>24.163 / 0.647</td>
</tr>
<tr>
<td>DBPN</td>
<td>2</td>
<td>29.512 / 0.827</td>
<td>26.371 / 0.711</td>
<td>25.268 / 0.662</td>
</tr>
<tr>
<td>DBPN + est. correction</td>
<td>2</td>
<td>27.784 / 0.828</td>
<td>27.761 / 0.769</td>
<td>27.103 / 0.722</td>
</tr>
</tbody>
</table>

to restore the HR image by (9). Note that Algorithm 1 exploits the pre-trained DNN to estimate the correction filter. Here we apply it only with DBPN, which has a more compact architecture than RCAN and proSR, and hence leads to faster inference. However, similar results can be obtained also for RCAN and proSR. In this setting we compare our method to kernelGAN [3], which estimates the downscaling kernel using adversarial training (in test-time) and then uses ZSSR to restore the HR image.

The results for Set14 and BSD100 are presented in Tables 4 and 5, respectively, and visual examples are shown in Figure 5. More results and comparison on DIV2KRK are presented in the supplementary material. It can be seen that the proposed filter correction approach improves the results of DBPN compared to its plain application. It also outperforms kernelGAN, despite being much simpler.

5. Conclusion

The SISR task has gained a lot from the developments in deep learning in the recent years. Yet, the leading DNN methods suffer from a huge performance drop when they are tested on images that do not fit the acquisition process assumption used in their training phase—which is, typically, that the downscaling kernel is bicubic. In this work, we addressed this issue by a signal processing approach: computing a correction filter that modifies the low-resolution observations such that they mimic observations that are obtained with a bicubic kernel. (Notice that our focus in this work on the bicubic kernel is for the sake of simplicity of the presentation and due to its popularity. Yet, it is possible to use our developed tools also for other reconstruction kernels). The modified LR is then fed into existing state-of-the-art DNNs that are trained only under the assumption of bicubic kernel. Various experiments have shown that the proposed approach significantly improves the performance of the pre-trained DNNs and outperforms other (much more sophisticated) methods that are specifically designed to be robust to different kernels.

Acknowledgment. The work is supported by the NSF-BSF (No. 2017729) and ERC-StG (No. 757497) grants.
References


