Unfolding the Alternating Optimization for Blind Super Resolution

Zhengxiong Luo\textsuperscript{1,2,3}, Yan Huang\textsuperscript{1,2}, Shang Li\textsuperscript{2,3}, Liang Wang\textsuperscript{1,4,5}, and Tieniu Tan\textsuperscript{1,4}

\textsuperscript{1} Center for Research on Intelligent Perception and Computing (CRIPAC)  
\textsuperscript{2} National Laboratory of Pattern Recognition (NLPR)  
\textsuperscript{3} Institute of Automation, Chinese Academy of Sciences (CASIA)  
\textsuperscript{4} School of Artificial Intelligence, University of Chinese Academy of Sciences (UCAS)  
\textsuperscript{5} Center for Excellence in Brain Science and Intelligence Technology (CEBSIT)  
\textsuperscript{5} Chinese Academy of Sciences, Artificial Intelligence Research (CAS-AIR)

Abstract

Previous methods decompose blind super resolution (SR) problem into two sequential steps: \( i \) estimating blur kernel from given low-resolution (LR) image and \( ii \) restoring SR image based on estimated kernel. This two-step solution involves two independently trained models, which may not be well compatible with each other. Small estimation error of the first step could cause severe performance drop of the second one. While on the other hand, the first step can only utilize limited information from LR image, which makes it difficult to predict highly accurate blur kernel. Towards these issues, instead of considering these two steps separately, we adopt an alternating optimization algorithm, which can estimate blur kernel and restore SR image in a single model. Specifically, we design two convolutional neural modules, namely \textit{Restorer} and \textit{Estimator}. \textit{Restorer} restores SR image based on predicted kernel, and \textit{Estimator} estimates blur kernel with the help of restored SR image. We alternate these two modules repeatedly and unfold this process to form an end-to-end trainable network. In this way, \textit{Estimator} utilizes information from both LR and SR images, which makes the estimation of blur kernel easier. More importantly, \textit{Restorer} is trained with the kernel estimated by \textit{Estimator}, instead of ground-truth kernel, thus \textit{Restorer} could be more tolerant to the estimation error of \textit{Estimator}. Extensive experiments on synthetic datasets and real-world images show that our model can largely outperform state-of-the-art methods and produce more visually favorable results at much higher speed. The source code is available at \url{https://github.com/greatlog/DAN.git}.

1 Introduction

Single image super resolution (SISR) aims to recover the high-resolution (HR) version of a given degraded low-resolution (LR) image. It has wide applications in video enhancement, medical imaging, as well as security and surveillance imaging. Mathematically, the degradation process can be expressed as

\[ y = (x \otimes k) \downarrow_s + n \]  

(1)

where \( x \) is the original HR image, \( y \) is the degraded LR image, \( \otimes \) denotes the two-dimensional convolution of \( x \) with blur kernel \( k \), \( n \) denotes Additive White Gaussian Noise (AWGN), and \( \downarrow_s \) denotes the standard \( s \)-fold downsampler, which means keeping only the upper-left pixel for each

\*Corresponding author

distinct $s \times s$ patch [35]. Then SISR refers to the process of recovering $x$ from $y$. It is a highly ill-posed problem due to this inverse property, and thus has always been a challenging task.

Recently, deep neural networks (DNNs) have achieved remarkable results on SISR. But most of these methods [39, 2, 40, 23, 8, 21] assume that the blur kernel is predefined as the kernel of bicubic interpolation. In this way, large number of training samples can be manually synthesized and further used to train powerful DNNs. However, blur kernels in real applications are much more complicated, and there is a domain gap between bicubically synthesized training samples and real images. This domain gap will lead to severe performance drop when these networks are applied to real applications. Thus, more attention should be paid to SR in the context of unknown blur kernels, i.e., blind SR.

In blind SR, there is one more undetermined variable, i.e., blur kernel $k$, and the optimization also becomes much more difficult. To make this problem easier to be solved, previous methods [27, 32, 38, 35] usually decompose it into two sequential steps: i) estimating blur kernel from LR image and ii) restoring SR image based on estimated kernel. This two-step solution involves two independently trained models, thus they may be not well compatible to each other. Small estimation error of the first step could cause severe performance drop of the following one [14]. But on the other hand, the first step can only utilize limited information from LR image, which makes it difficult to predict highly accurate blur kernel. As a result, although both models can perform well individually, the final result may be suboptimal when they are combined together.

Instead of considering these two steps separately, we adopt an alternating optimization algorithm, which can estimate blur kernel $k$ and restore SR image $x$ in the same model. Specifically, we design two convolutional neural modules, namely Restorer and Estimator. Restorer restores SR image based on blur kernel predicted by Estimator, and the restored SR image is further used to help Estimator estimate better blur kernel. Once the blur kernel is manually initialized, the two modules can well cooperate with each other to form a closed loop, which can be iterated over and over. The iterating process is then unfolded to an end-to-end trainable network, which is called deep alternating network (DAN). In this way, Estimator can utilize information from both LR and SR images, which makes the estimation of blur kernel easier. More importantly, Restorer is trained with the kernel estimated by Estimator, instead of ground-truth kernel. Thus during testing Restorer could be more tolerant to the estimation error of Estimator. Besides, the results of both modules could be substantially improved during the iterations, thus it is likely for our alternating optimization algorithm to get better final results than the direct two-step solutions. We summarize our contributions into three points:

1. We adopt an alternating optimization algorithm to estimate blur kernel and restore SR image for blind SR in a single network (DAN), which helps the two modules to be well compatible with each other and likely to get better final results than previous two-step solutions.

2. We design two convolutional neural modules, which can be alternated repeatedly and then unfolded to form an end-to-end trainable network, without any pre/post-processing. It is easier to be trained and has higher speed than previous two-step solutions. To the best of our knowledge, the proposed method is the first end-to-end network for blind SR.

3. Extensive experiments on synthetic datasets and real-world images show that our model can largely outperform state-of-the-art methods and produce more visually favorable results at much higher speed.

## 2 Related Work

### 2.1 Super Resolution in the Context of Bicubic Interpolation

Learning based methods for SISR usually require a large number of paired HR and LR images as training samples. However, these paired samples are hard to get in real world. As a result, researchers manually synthesize LR images from HR images with predefined downsampling settings. The most popular setting is bicubic interpolation, i.e., defining $k$ in Equation 1 as bicubic kernel. From the arising of SRCNN [9], various DNNs [21, 40, 39, 10, 16, 18] have been proposed based on this setting. Recently, after the proposal of RCAN [39] and RRDB [30], the performance of these non-blind methods even start to saturate on common benchmark datasets. However, the blur kernels for real images are indeed much more complicated. In real applications, kernels are unknown and differ from image to image. As a result, despite that these methods have excellent performance in the context of bicubic downsampling, they still cannot be directly applied to real images due to the domain gap.
2.2 Super Resolution for Multiple Degradations

Another kind of non-blind SR methods aims to propose a single model for multiple degradations, i.e., the second step of the two-step solution for blind SR. These methods take both LR image and its corresponding blur kernel as inputs. In [13, 29], the blur kernel is used to downsample images and synthesize training samples, which can be used to train a specific model for given kernel and LR image. In [37], the blur kernel and LR image are directly concatenated at the first layer of a DNN. Thus, the SR result can be closely correlated to both LR image and blur kernel. In [38], Zhang et al. proposed a method based on ADMM algorithm. They interpret this problem as MAP optimization and solve the data term and prior term alternately. In [14], a spatial feature transform (SFT) layer is proposed to better preserve the details in LR image while blur kernel is an additional input. However, as pointed out in [14], the SR results of these methods are usually sensitive to the provided blur kernels. Small deviation of provided kernel from the ground truth will cause severe performance drop of these non-blind SR methods.

2.3 Blind Super Resolution

Previous methods for blind SR are usually the sequential combinations of a kernel-estimation method and a non-blind SR method. Thus kernel-estimation methods are also an important part of blind SR. In [26], Michaeli et al. estimate the blur kernel by utilizing the internal patch recurrence. In [3] and [5], LR image is firstly downsampled by a generative network, and then a discriminator is used to verify whether the downsampled image has the same distribution with original LR image. In this way, the blur kernel can be learned by the generative network. In [14], Gu et al. not only train a network for kernel estimation, but also propose a correction network to iteratively correct the kernel. Although the accuracy of estimated kernel is largely improved, it requires training of two or even three networks, which is rather complicated. Instead, DAN is an end-to-end trainable network that is much easier to be trained and has much higher speed.

3 End-to-End Blind Super Resolution

3.1 Problem Formulation

As shown in Equation[1] there are three variables, i.e., $x$, $k$, and $n$, to be determined in blind SR problem. In literature, we can apply a denoise algorithm [36, 7, 15] in the first place. Then blind SR algorithm only needs to focus on solving $k$ and $x$. It can be mathematically expressed an optimization problem:

$$\arg \min_{k, x} \| y - (x \otimes k) \downarrow_s \|_1 + \phi(x)$$

(2)

where the former part is the reconstruction term, and $\phi(x)$ is prior term for HR image. The prior term is usually unknown and hard to be analytically expressed. Thus it is extremely difficult to solve this problem directly. Previous methods decompose this problem into two sequential steps:

$$\begin{cases} k = M(y) \\ x = \arg \min_{x} \| y - (x \otimes k) \downarrow_s \|_1 + \phi(x) \end{cases}$$

(3)

where $M(\cdot)$ denotes the function that estimates $k$ from $y$, and the second step is usually solved by a non-blind SR method described in Sec 2.2. This two-step solution has its drawbacks in
threefold. Firstly, this algorithm usually requires training of two or even more models, which is rather complicated. Secondly, $M(\cdot)$ can only utilize information from $y$, which treats $k$ as a kind of prior of $y$. But in fact, $k$ could not be properly solved without information from $x$. At last, the non-blind SR model for the second step is trained with ground-truth kernels. While during testing, it can only have access to kernels estimated in the first step. The difference between ground-truth and estimated kernels will usually cause serve performance drop of the non-blind SR model [14].

Towards these drawbacks, we propose an end-to-end network that can largely release these issues. We still split it into two subproblems, but instead of solving them sequentially, we adopt an alternating optimization algorithm, which restores SR image and estimates corresponding blur kernel alternately. The mathematical expression is

$$\begin{align*}
\textbf{k}_{t+1} & = \arg \min_{k} \| y - (x \otimes k) \downarrow s \|_1 \\
\textbf{x}_{t+1} & = \arg \min_{x} \| y - (x \otimes \textbf{k}_t) \downarrow s \|_1 + \phi(x)
\end{align*}$$

We alternately solve these two subproblems both via convolutional neural modules, namely Estimator and Restorer respectively. Actually, there even has an analytic solution for Estimator. But we experimentally find that analytic solution is more time-consuming and not robust enough (when noise is not fully removed). We fix the number of iterations as $T$ and unfold the iterating process to form an end-to-end trainable network, which is called deep alternating network (DAN).

As shown in Figure [1] we initialize the kernel by Dirac function, i.e., the center of the kernel is one and zeros otherwise. Following [14] [37], the kernel is also reshaped and then reduced by principal component analysis (PCA). We set $T = 4$ in practice and both modules are supervised only at the last iteration by L1 loss. The whole network could be well trained without any restrictions on intermediate results, because the parameters of both modules are shared between different iterations.

In DAN, Estimator takes both LR and SR images as inputs, which makes the estimation of blur kernel $k$ much easier. More importantly, Restorer is trained with the kernel estimated by Estimator, instead of ground-truth kernel as previous methods do. Thus, Restorer could be more tolerant to the estimation error of Estimator during testing. Besides, compared with previous two-step solutions, the results of both modules in DAN could be substantially improved, and it is likely for DAN to get better final results. Specially, in the case where the scale factor $s = 1$, DAN becomes an deblurring network. Due to limited pages, we only discuss SR cases in this paper.

### 3.2 Instantiate the Convolutional Neural Modules

Both modules in our network have two inputs. Estimator takes LR and SR image, and Restorer takes LR image and blur kernel as inputs. We define LR image as basic input, and the other one is conditional input. For example, blur kernel is the conditional input of Restorer. During iterations, the basic inputs of both modules keep the same, but their conditional inputs are repeatedly updated. We claim that it is significantly important to keep the output of each module closely related to its conditional input. Otherwise, the iterating results will collapse to a fixed point at the first iteration. Specifically, if Estimator outputs the same kernel regardless of the value of SR image, or Restorer outputs the same SR image regardless of the value of blur kernel, their outputs will only depend on the basic input, and the results will keep the same during the iterations.

To ensure that the outputs of Estimator and Restorer are closely related to their conditional inputs, we propose a conditional residual block (CRB). On the basis of the residual block in [39], we concatenate the conditional and basic inputs at the beginning:

$$f_{out} = R(\text{Concat}(f_{\text{basic}}, f_{\text{cond}})) + f_{\text{basic}}$$

where $R(\cdot)$ denotes the residual mapping function of CRB and $\text{Concat}(\cdot, \cdot)$ denotes concatenation. $f_{\text{basic}}$ and $f_{\text{cond}}$ are the basic input and conditional input respectively. As shown in Figure [2](c), the residual mapping function consists of two $3 \times 3$ convolutional layers and one channel attention layer [17]. Both Estimator and Restorer are build by CRBs.

**Estimator.** The whole structure of Estimator is shown in Figure [2](b). We firstly downsample SR image by a convolutional layer with stride $s$. Then the feature maps are sent to all CRBs as conditional inputs. At the end of the network, we squeeze the features by global average pooling to form the elements of predicted kernel. Since the kernel is reduced by PCA, Estimator only needs to estimate
the PCA result of blur kernel. In practice, Estimator has 5 CRBs, and both basic input and conditional input of each CRB have 32 channels.

Resicter. The whole structure of Resticer is shown in Figure 2(a). In Resticer, we stretch the kernel in spatial dimension to the same spatial size as LR image. Then the stretched kernel is sent to all CRBs of Resticer as conditional inputs. We use PixelShuffle [28] layers to upscale the features to desired size. In practice, Resticer has 40 CRBs, and the basic input and conditional input of each CRB has 64 and 10 channels respectively.

4 Experiments

4.1 Experiments on Synthetic Test Images

4.1.1 Data, Training and Testing

We collect 3450 HR images from DIV2K [1] and Flickr2K [11] as training set. To make reasonable comparison with other methods, we train models with two different degradation settings. One is the setting in [14], which only focuses on cases with isotropic Gaussian blur kernels. The other is the setting in [3], which focuses on cases with more general and irregular blur kernels.

Setting 1. Following the setting in [14], the kernel size is set as 21. During training, the kernel width is uniformly sampled in [0.2, 4.0], [0.2, 3.0] and [0.2, 2.0] for scale factors 4, 3 and 2 respectively. For quantitative evaluation, we collect HR images from the commonly used benchmark datasets, i.e. Set5 [4], Set14 [34], Urban100 [19], BSD100 [24] and Manga109 [25]. Since determined kernels are needed for reasonable comparison, we uniformly choose 8 kernels, denoted as Gaussian8, from range [1.8, 3.2], [1.35, 2.40] and [0.80, 1.60] for scale factors 4, 3 and 2 respectively. The HR images are first blurred by the selected blur kernels and then downsampled to form synthetic test images.

Setting 2. Following the setting in [3], we set the kernel size as 11. We firstly generate anisotropic Gaussian kernels. The lengths of both axises are uniformly distributed in (0.6, 5), rotated by a random angle uniformly distributed in [−π, π]. To deviate from a regular Gaussian, we further apply uniform multiplicative noise (up to 25% of each pixel value of the kernel) and normalize it to sum to one. For testing, we use the benchmark dataset DIV2KRK that is used in [3].

The input size during training is 64 × 64 for all scale factors. The batch size is 64. Each model is trained for 4 × 10^5 iterations. We use Adam [22] as our optimizer, with β1 = 0.9, β2 = 0.99. The initial learning rate is 2 × 10^{-4}, and will decay by half at every 1 × 10^5 iterations. All models are trained on RTX2080Ti GPUs.
Table 1: Quantitative comparison with SOTA SR methods with Setting 1. The best two results are highlighted in red and blue colors respectively.

<table>
<thead>
<tr>
<th>Method</th>
<th>Scale</th>
<th>Set5 PSNR</th>
<th>Set5 SSIM</th>
<th>Set14 PSNR</th>
<th>Set14 SSIM</th>
<th>BSD100 PSNR</th>
<th>BSD100 SSIM</th>
<th>Urban100 PSNR</th>
<th>Urban100 SSIM</th>
<th>Manga109 PSNR</th>
<th>Manga109 SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bicubic</td>
<td>2</td>
<td>28.82</td>
<td>0.8577</td>
<td>26.02</td>
<td>0.7879</td>
<td>25.92</td>
<td>0.7310</td>
<td>23.14</td>
<td>0.7089</td>
<td>25.80</td>
<td>0.8949</td>
</tr>
<tr>
<td>CARN [2]</td>
<td>2</td>
<td>30.99</td>
<td>0.8779</td>
<td>28.10</td>
<td>0.8789</td>
<td>26.78</td>
<td>0.7286</td>
<td>25.27</td>
<td>0.7630</td>
<td>26.86</td>
<td>0.8806</td>
</tr>
<tr>
<td>ZSSR [9]</td>
<td>2</td>
<td>31.08</td>
<td>0.8786</td>
<td>28.35</td>
<td>0.7933</td>
<td>27.92</td>
<td>0.7632</td>
<td>25.25</td>
<td>0.7618</td>
<td>28.05</td>
<td>0.8769</td>
</tr>
<tr>
<td>KIC [14]</td>
<td>2</td>
<td>32.10</td>
<td>0.8658</td>
<td>32.82</td>
<td>0.8999</td>
<td>31.36</td>
<td>0.9097</td>
<td>30.36</td>
<td>0.8949</td>
<td>36.06</td>
<td>0.9474</td>
</tr>
<tr>
<td>DAN</td>
<td>2</td>
<td>32.71</td>
<td>0.8974</td>
<td>31.27</td>
<td>0.8267</td>
<td>28.72</td>
<td>0.8033</td>
<td>25.62</td>
<td>0.7981</td>
<td>29.58</td>
<td>0.9134</td>
</tr>
<tr>
<td>CARN [2]</td>
<td>3</td>
<td>26.21</td>
<td>0.7766</td>
<td>24.01</td>
<td>0.6662</td>
<td>24.25</td>
<td>0.6356</td>
<td>21.39</td>
<td>0.6203</td>
<td>22.98</td>
<td>0.7576</td>
</tr>
<tr>
<td>ZSSR [9]</td>
<td>3</td>
<td>28.25</td>
<td>0.7989</td>
<td>26.11</td>
<td>0.6942</td>
<td>26.06</td>
<td>0.6633</td>
<td>23.26</td>
<td>0.6534</td>
<td>25.19</td>
<td>0.7914</td>
</tr>
<tr>
<td>KIC [14]</td>
<td>3</td>
<td>30.31</td>
<td>0.8562</td>
<td>27.57</td>
<td>0.7531</td>
<td>27.14</td>
<td>0.7152</td>
<td>24.45</td>
<td>0.7241</td>
<td>27.67</td>
<td>0.8592</td>
</tr>
<tr>
<td>DAN</td>
<td>3</td>
<td>32.16</td>
<td>0.9420</td>
<td>29.46</td>
<td>0.8229</td>
<td>28.56</td>
<td>0.8493</td>
<td>25.94</td>
<td>0.8165</td>
<td>28.21</td>
<td>0.8739</td>
</tr>
</tbody>
</table>

Table 4.1: Visual results of img 005 in Urban100. The width of blur kernel is 1.8.

4.1.2 Quantitative Results

**Setting 1.** For the first setting, we evaluate our method on test images synthesized by Gaussian8 kernels. We mainly compare our results with ZSSR [29] and IKC [14], which are methods designed for blind SR. We also include a comparison with CARN [2]. Since it is not designed for blind SR, we perform deblurring method [27] before or after CARN. The PSNR and SSIM results on Y channel of transformed YCbCr space are shown in Table 1.

Despite that CARN achieves remarkable results in the context of bicubic downsampling, it suffers severe performance drop when applied to images with unknown blur kernels. Its performance is largely improved when it is followed by a deblurring method, but still inferior to that of blind-SR methods. ZSSR trains specific network for each single tested image by utilizing the internal patch recurrence. However, ZSSR has an in-born drawback: the training samples for each image are limited, and thus it cannot learn a good prior for HR images. IKC is also a two-step solution for blind SR. Although the accuracy of estimated kernel is largely improved in IKC, the final result is still suboptimal. DAN is trained in an end-to-end scheme, which is not only much easier to be trained than two-step solutions, but also likely to a reach a better optimum point. As shown in Table 1, the PSNR result of DAN on Manga109 for scale ×3 is even 4.95dB higher than that of IKC. For other scales and datasets, DAN also largely outperforms IKC.

The visual results of img 005 in Urban100 are shown in Figure 3 for comparison. As one can see, CARN and ZSSR even cannot restore the edges for the window. IKC performs better, but the edges are severely blurred. While DAN can restore sharp edges and produce more visually pleasant result.

**Setting 2.** The second setting involves irregular blur kernels, which is more general, but also more difficult to solve. For Setting 2, we mainly compare methods of three different classes: i) SOTA SR algorithms trained on bicubically downsampled images such as EDSR [23] and RCAN [39], ii) blind SR methods designed for NTIRE competition such as PDN [31] and WDSR [33], iii) the two-step solutions, i.e. the combination of a kernel estimation method and a non-blind SR method, such as Kernel-GAN [3] and ZSSR [29]. The PSNR and SSIM results on Y channel are shown in Table 2.

Similarly, the performance of methods trained on bicubically downsampled images is limited by the domain gap. Thus, their results are only slightly better than that of interpolation. The methods in Class 2 are trained on synthesized images provided in NTIRE competition. Although these methods
Table 2: Quantitative comparison with SOTA SR methods with Setting 2. The best two results are highlighted in red and blue colors respectively.

<table>
<thead>
<tr>
<th>Types</th>
<th>Method</th>
<th>Scale</th>
<th>PSNR</th>
<th>SSIM</th>
<th>PSNR</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class 1</td>
<td>Bicubic</td>
<td></td>
<td>28.73</td>
<td>0.8040</td>
<td>25.33</td>
<td>0.6795</td>
</tr>
<tr>
<td></td>
<td>Bicubic kernel + ZSSR</td>
<td>29.10</td>
<td>0.8215</td>
<td>25.61</td>
<td>0.6911</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EDSR [23]</td>
<td>29.17</td>
<td>0.8216</td>
<td>25.64</td>
<td>0.6928</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RCAN [39]</td>
<td>29.20</td>
<td>0.8223</td>
<td>25.66</td>
<td>0.6936</td>
<td></td>
</tr>
<tr>
<td>Class 2</td>
<td>PDN [31] - 1st in NTIRE’19 track4</td>
<td></td>
<td>/</td>
<td>/</td>
<td>26.34</td>
<td>0.7190</td>
</tr>
<tr>
<td></td>
<td>WDSR [33] - 1st in NTIRE’19 track2</td>
<td></td>
<td>/</td>
<td>/</td>
<td>21.55</td>
<td>0.6841</td>
</tr>
<tr>
<td></td>
<td>WDSR [33] - 1st in NTIRE’19 track3</td>
<td></td>
<td>/</td>
<td>/</td>
<td>21.54</td>
<td>0.7016</td>
</tr>
<tr>
<td></td>
<td>WDSR [33] - 2nd in NTIRE’19 track4</td>
<td></td>
<td>/</td>
<td>/</td>
<td>25.64</td>
<td>0.7144</td>
</tr>
<tr>
<td></td>
<td>Ji et al. [20] - 1st in NITRE’20 track 1</td>
<td></td>
<td>/</td>
<td>/</td>
<td>25.43</td>
<td>0.6907</td>
</tr>
<tr>
<td>Class 3</td>
<td>Cornillere et al. [6]</td>
<td></td>
<td>29.46</td>
<td>0.8474</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>Michaeli et al. [26] + SRMD [37]</td>
<td></td>
<td>25.51</td>
<td>0.8083</td>
<td>23.34</td>
<td>0.6530</td>
</tr>
<tr>
<td></td>
<td>Michaeli et al. [26] + ZSSR [29]</td>
<td></td>
<td>29.37</td>
<td>0.8370</td>
<td>26.09</td>
<td>0.7138</td>
</tr>
<tr>
<td></td>
<td>KernelGAN [3] + SRMD [37]</td>
<td></td>
<td>29.57</td>
<td>0.8564</td>
<td>25.71</td>
<td>0.7265</td>
</tr>
<tr>
<td></td>
<td>KernelGAN [3] + USRNet [35]</td>
<td></td>
<td>20.06</td>
<td>0.5359</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>KernelGAN [3] + ZSSR [29]</td>
<td></td>
<td>30.36</td>
<td>0.8669</td>
<td>26.81</td>
<td>0.7316</td>
</tr>
<tr>
<td></td>
<td>Ours DAN</td>
<td></td>
<td>32.56</td>
<td>0.8997</td>
<td>27.55</td>
<td>0.7582</td>
</tr>
</tbody>
</table>

Figure 4: Visual results on img 892 in DIV2KRK.

Table 3: PSNR results when GT kernels are provided.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Set5</th>
<th>Set14</th>
<th>B100</th>
<th>Urban100</th>
<th>Manga109</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAN</td>
<td>31.89</td>
<td>28.43</td>
<td>27.51</td>
<td>25.86</td>
<td>30.50</td>
</tr>
<tr>
<td>DAN(GT kernel)</td>
<td>31.85</td>
<td>28.42</td>
<td>27.51</td>
<td>25.87</td>
<td>30.51</td>
</tr>
</tbody>
</table>

4.1.3 Study of Estimated Kernels

To evaluated the accuracy of predicted kernels, we calculate their L1 errors in the reduced space, and the results on Urban100 are shown in Figure 5(a). As one can see that the L1 error of reduced kernels predicted by DAN are much lower than that of IKC. It suggests that the overall improvements of DAN may partially come from more accurate retrieved kernels. We also plot the PSNR results with respect to kernels with different sigma in Figure 5(b). As sigma increases, the performance gap
between IKC and DAN also becomes larger. It indicates that DAN may have better generalization ability.

We also replace the estimated kernel by ground truth (GT) to further investigate the influence of Estimator. If GT kernels are provided, the iterating processing becomes meaningless. Thus we test the Restorer with just once forward propagation. The tested results for Setting 1 is shown in Table 3. The result almost keeps unchanged and sometimes even gets worse when GT kernels are provided. It indicates that Predictor may have already satisfied the requirements of Restorer, and the superiority of DAN also partially comes from the good cooperation between its Predictor and Restorer.

### 4.1.4 Study of Iterations

After the model is trained, we also change the number of iterations to see whether the two modules have learned the property of convergence or just have ‘remembered’ the iteration number. The model is trained with 4 iterations, but during testing we increase the iteration number from 1 to 7. As shown in Figure 6(a) and (c), the average PSNR results on Set5 and Set14 firstly increase rapidly and then gradually converge. It should be noted that when we iterate more times than training, the performance dose not becomes worse, and sometimes even becomes better. For example, the average PSNR on Set14 is 20.43 dB when the iteration number is 5, higher than 20.42 dB when we iterate 4 times. Although the incremental is relatively small, it suggests that the two modules may have learned to cooperate with each other, instead of solving this problem like ordinary end-to-end networks, in which cases, the performance will drop significantly when the setting of testing is different from that of training. It also suggests that the estimation error of intermediate results does not destroy the convergence of DAN. In other words, DAN is robust to various estimation error.

### 4.2 Inference Speed

One more superiority of our end-to-end model is that it has higher inference speed. To make a quantitative comparison, we evaluate the average speed of different methods on the same platform. We choose the 40 images synthesized by Gaussian kernels from Set5 as testing images, and all methods are evaluated on the same platform with a RTX2080Ti GPU. We choose KernelGAN [3] + ZSSR [29] as the one of the representative methods. Its speed is 415.7 seconds per image. IKC [14] has much faster inference speed, which is only 3.93 seconds per image. As a comparison, the average speed of DAN is 0.75 seconds per image, nearly 554 times faster than KernelGAN + ZSSR, and 5 times faster than IKC. In other words, DAN not only can largely outperform SOTA blind SR methods on PSNR results, but also has much higher speed.

### 4.3 Experiments on Real World Images

We also conduct experiments to prove that DAN can generalize well to real world images. In this case, we need to consider the influence of additive noise. As we mentioned in Sec 3.1, we can perform an denoise algorithm in the first place. But for simplicity, we retrain a different model by adding AWGN to LR image during training. In this way, DAN would be forced to generalize to noisy images.
The covariance of noise is set as $15$. We use KernelGAN [3] + ZSSR [29] and IKC [14] as the representative methods for blind SR, and CARN [2] as the representative method for non-blind SR method. The commonly used image chip [12] is chosen as test image. It should be noted that it is a real image and we do not have the ground truth. Thus we can only provide a visual comparison in Figure [12]. As one can see, the result of KernelGAN + ZSSR is slightly better than bicubic interpolation, but is still heavily blurred. The result of CARN is over smoothed and the edge is not sharp enough. IKC produces cleaner result, but there are still some artifacts. The letter ‘X’ restored by IKC has an obvious dark line at the top right part. But this dark line is much lighter in the image restored by DAN. It suggests that if trained with noisy images, DAN can also learn to denoise, and produce more visually pleasant results with more reliable details. This is because that both modules are implemented via convolutional layers, which are flexible enough to be adapted to different tasks.

Figure 6: PSNR and visual results with different iterations during testing on Set5 and Set14.

5 Conclusion

In this paper, we have proposed an end-to-end algorithm for blind SR. This algorithm is based on alternating optimization, the two parts of which are both implemented by convolutional modules, namely Restorer and Estimator. We unfold the alternating process to form an end-to-end trainable network. In this way, Estimator can utilize information from both LR and SR images, which makes it easier to estimate blur kernel. More importantly, Restorer is trained with the kernel estimated by Estimator, instead of ground-truth kernel, thus Restorer could be more tolerant to with the estimation error of Estimator. Besides, the results of both modules could be substantially improved during the iterations, thus it is likely for DAN to get better final results than previous two-step solutions. Experiments also prove that DAN outperforms SOTA blind SR methods by a large margin. In the future, if the two parts of DAN can be implemented by more powerful modules, we believe that its performance could be further improved.

Broader Impact

Super Resolution is a traditional task in computer vision. It has been studied for several decades and has wide applications in video enhancement, medical imaging, as well as security and surveillance imaging. These techniques have largely benefited the society in various areas for years and have
no negative impact yet. The proposed method (DAN) could further improve the merits of these applications especially in cases where the degradations are unknown. DAN has relatively better performance and much higher speed, and it is possible for DAN to be used in real-time video enhancement or surveillance imaging. This work does not present any negative foreseeable societal consequence.

Acknowledgements

This work is jointly supported by National Key Research and Development Program of China (2016YFB1001000), Key Research Program of Frontier Sciences, CAS (ZDBS-LY-JSC032), Shandong Provincial Key Research and Development Program (2019JZZY010119), and CAS-AIR.

References


