arXiv:2010.02631v3 [cs.CV] 16 Oct 2020

Unfolding the Alternating Optimization for Blind Super Resolution

Zhengxiong Luo[1,2,3], *Yan Huang[1,2], Shang Li[2,3], Liang Wang[1,4,5], and Tieniu Tan[1,4]

[1] Center for Research on Intelligent Perception and Computing (CRIPAC)
National Laboratory of Pattern Recognition (NLPR)
[2] Institute of Automation, Chinese Academy of Sciences (CASIA)
[3] School of Artificial Intelligence, University of Chinese Academy of Sciences (UCAS)
[4] Center for Excellence in Brain Science and Intelligence Technology (CEBSIT)
[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 *Restorer* and *Estimator*. *Restorer* restores SR image based on predicted kernel, and *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, *Estimator* utilizes 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 *Restorer* could be more tolerant to the estimation error of *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 `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

$$\mathbf{y} = (\mathbf{x} \otimes \mathbf{k}) \downarrow_s + \mathbf{n} \tag{1}$$

where $\mathbf{x}$ is the original HR image, $\mathbf{y}$ is the degraded LR image, $\otimes$ denotes the two-dimensional convolution of $\mathbf{x}$ with blur kernel $\mathbf{k}$, $\mathbf{n}$ 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

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

distinct $s \times s$ patch [35]. Then SISR refers to the process of recovering $\mathbf{x}$ from $\mathbf{y}$. 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 $\mathbf{k}$, and the optimization also becomes much more difficult. To make this problem easier to be solved, previous methods [37, 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 $\mathbf{k}$ and restore SR image $\mathbf{x}$ 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 corporate 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 $\mathbf{k}$ 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.

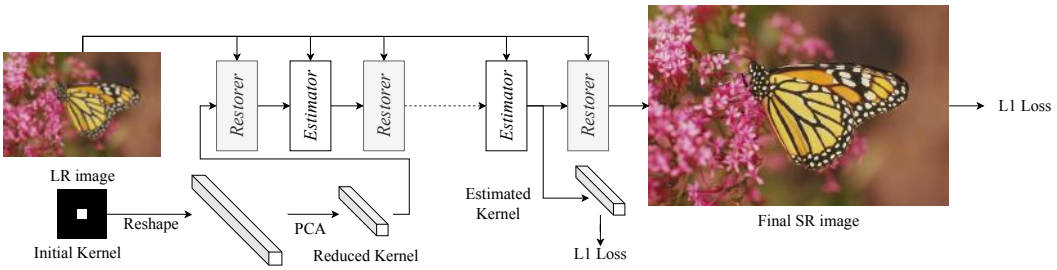


Figure 1: The overview structure of the deep alternating network (DAN).

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*. $\mathbf{x}$, $\mathbf{k}$ and $\mathbf{n}$, 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 $\mathbf{k}$ and $\mathbf{x}$. It can be mathematically expressed an optimization problem:

$$\arg\min_{\mathbf{k},\mathbf{x}} \|\mathbf{y} - (\mathbf{x} \otimes \mathbf{k}) \downarrow_s \|_1 + \phi(\mathbf{x}) \tag{2}$$

where the former part is the reconstruction term, and $\phi(\mathbf{x})$ is prior term for HR image. The prior term is usually unknown and has no analytic expression. Thus it is extremely difficult to solve this problem directly. Previous methods decompose this problem into two sequential steps:

$$\begin{cases} \mathbf{k} = M(\mathbf{y}) \\ \mathbf{x} = \arg\min_{\mathbf{x}} \|\mathbf{y} - (\mathbf{x} \otimes \mathbf{k}) \downarrow_s \|_1 + \phi(\mathbf{x}) \end{cases} \tag{3}$$

where $M(\cdot)$ denotes the function that estimates $\mathbf{k}$ from $\mathbf{y}$, 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 $\mathbf{y}$, which treats $\mathbf{k}$ as a kind of prior of $\mathbf{y}$. But in fact, $\mathbf{k}$ could not be properly solved without information from $\mathbf{x}$. 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

$$\begin{cases} \mathbf{k}_{i+1} = \arg\min_{\mathbf{k}} \|\mathbf{y} - (\mathbf{x}_i \otimes \mathbf{k}) \downarrow_s \|_1 \\ \mathbf{x}_{i+1} = \arg\min_{\mathbf{x}} \|\mathbf{y} - (\mathbf{x} \otimes \mathbf{k}_i) \downarrow_s \|_1 + \phi(\mathbf{x}) \end{cases} \tag{4}$$

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 $\mathbf{k}$ 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 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:

$$f_{out} = R(Concat([f_{basic}, f_{cond}])) + f_{basic} \tag{5}$$

where $R(\cdot)$ denotes the residual mapping function of CRB and $Concat([\cdot, \cdot])$ denotes concatenation. f_{basic} and f_{cond} are the basic input and conditional input respectively. As shown in Figure 2 (c), the residual mapping function consists of two 3×3 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

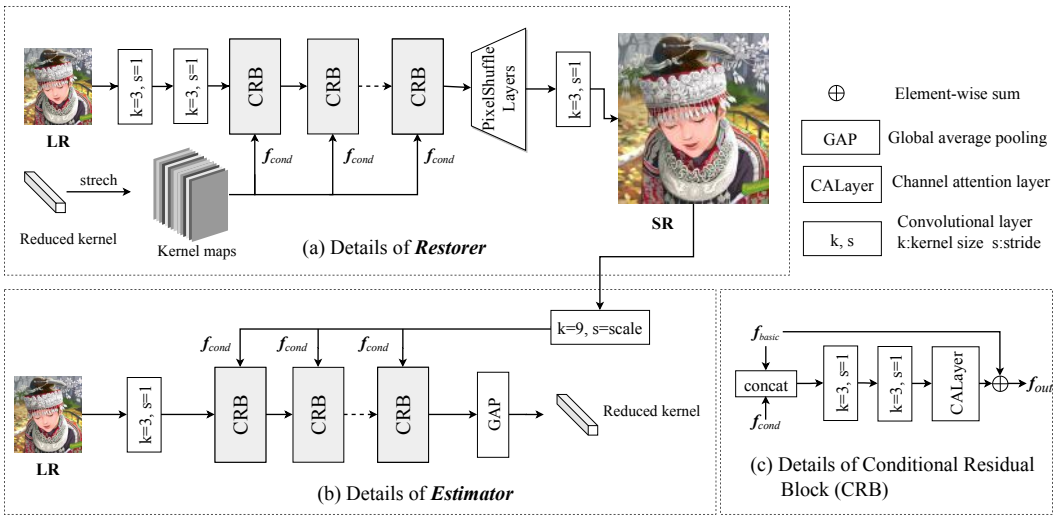


Figure 2: The details of *Estimator* and *Restorer*. (a) (top) The details of *Restorer*. (c) (bottom-left) The details of *Estimator*. (c) (bottom-right) The details of conditional residual block (CRB).

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.

Restorer. The whole structure of *Restorer* is shown in Figure 2 (a). In *Restorer*, 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 *Restorer* as conditional inputs. We use PixelShuffle [28] layers to upscale the features to desired size. In practice, *Restorer* 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 $[-\pi, \pi]$. 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 $\beta_1 = 0.9$, $\beta_2 = 0.99$. 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.

Method	Scale	Set5		Set14		BSD100		Urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	2	28.82	0.8577	26.02	0.7634	25.92	0.7310	23.14	0.7258	25.60	0.8498
CARN [2]		30.99	0.8779	28.10	0.7879	26.78	0.7286	25.27	0.7630	26.86	0.8606
ZSSR [29]		31.08	0.8786	28.35	0.7933	27.92	0.7632	25.25	0.7618	28.05	0.8769
[27]+CARN [2]		24.20	0.7496	21.12	0.6170	22.69	0.6471	18.89	0.5895	21.54	0.7946
CARN [2]+[27]		31.27	0.8974	29.03	0.8267	28.72	0.8033	25.62	0.7981	29.58	0.9134
IKC [14]		36.62	0.9658	32.82	0.8999	31.36	0.9097	30.36	0.8949	36.06	0.9474
DAN		37.33	0.9754	33.07	0.9068	31.76	0.9213	30.60	0.9020	37.23	0.9710
Bicubic	3	26.21	0.7766	24.01	0.6662	24.25	0.6356	21.39	0.6203	22.98	0.7576
CARN [2]		27.26	0.7855	25.06	0.6676	25.85	0.6566	22.67	0.6323	23.84	0.7620
ZSSR [29]		28.25	0.7989	26.11	0.6942	26.06	0.6633	23.26	0.6534	25.19	0.7914
[27]+CARN [2]		19.05	0.5226	17.61	0.4558	20.51	0.5331	16.72	0.4578	18.38	0.6118
CARN [2]+[27]		30.31	0.8562	2757	0.7531	27.14	0.7152	24.45	0.7241	27.67	0.8592
IKC [14]		32.16	0.9420	29.46	0.8229	28.56	0.8493	25.94	0.8165	28.21	0.8739
DAN		34.04	0.9620	30.09	0.8287	28.94	0.8606	27.65	0.8352	33.16	0.9382
Bicubic	4	24.57	0.7108	22.79	0.6032	23.29	0.5786	20.35	0.5532	21.50	0.6933
CARN [2]		26.57	0.7420	24.62	0.6226	24.79	0.5963	22.17	0.5865	21.85	0.6834
ZSSR [29]		26.45	0.7279	24.78	0.6268	24.97	0.5989	22.11	0.5805	23.53	0.7240
[27]+CARN [2]		18.10	0.4843	16.59	0.3994	18.46	0.4481	15.47	0.3872	16.78	0.5371
CARN [2]+[27]		28.69	0.8092	26.40	0.6926	26.10	0.6528	23.46	0.6597	25.84	0.8035
IKC [14]		31.52	0.9278	28.26	0.7688	27.29	0.8041	25.33	0.7760	29.90	0.8793
DAN		31.89	0.9302	28.43	0.7693	27.51	0.8078	25.86	0.7822	30.50	0.9037

Figure 3: 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 $\times 3$ 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 channl 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.

Types	Method	Scale 2		4	
		PSNR	SSIM	PSNR	SSIM
Class 1	Bicubic	28.73	0.8040	25.33	0.6795
	Bicubic kernel + ZSSR [29]	29.10	0.8215	25.61	0.6911
	EDSR [23]	29.17	0.8216	25.64	0.6928
	RCAN [39]	29.20	0.8223	25.66	0.6936
Class 2	PDN [31] - 1st in NTIRE'19 track4	/	/	26.34	0.7190
	WDSR [33] - 1st in NTIIRE'19 track2	/	/	21.55	0.6841
	WDSR [33] - 1st in NTIRE'19 track3	/	/	21.54	0.7016
	WDSR [33] - 2nd in NTIRE'19 track4	/	/	25.64	0.7144
	Ji *et al.* [20] - 1st in NITRE'20 track 1	/	/	25.43	0.6907
Class 3	Cornillere *et al.* [6]	29.46	0.8474	/	/
	Michaeli *et al.* [26] + SRMD [37]	25.51	0.8083	23.34	0.6530
	Michaeli *et al.* [26] + ZSSR [29]	29.37	0.8370	26.09	0.7138
	KernelGAN [3] + SRMD [37]	29.57	0.8564	25.71	0.7265
	KernelGAN [3] + USRNet [35]	/	/	20.06	0.5359
	KernelGAN [3]+ ZSSR [29]	30.36	0.8669	26.81	0.7316
Ours	DAN	32.56	0.8997	27.55	0.7582

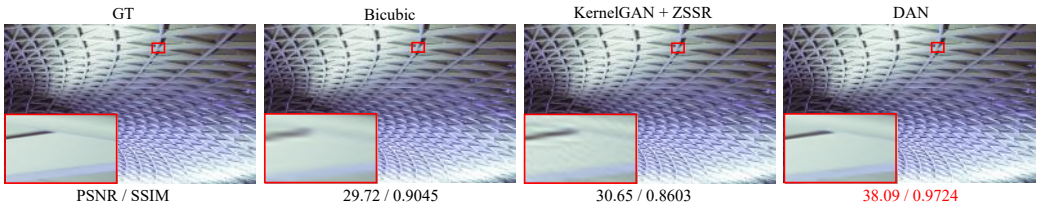


Figure 4: Visual results on *img 892* in DIV2KRK.

achieve remarkable results in the competition, they still cannot generalize well to irregular blur kernels.

The comparison between methods of Class 3 can enlighten us a lot. Specifically, USRNet [35] achieves remarkable results when GT kernels are provided, and KernelGAN also performs well on kernel estimation. However, when they are combined together, as shown in Table 2, the final SR results are worse than all other methods. This indicates that it is important for the *Estimator* and *Restorer* to be compatible with each other. Additionally, although better kernel-estimation method can benefit the SR results, the overall performance is still largely inferior to that of DAN. DAN outperforms the combination of KernelGAN and ZSSR by $2.20dB$ and $0.74dB$ for scales $\times 2$ and $\times 4$ respectively.

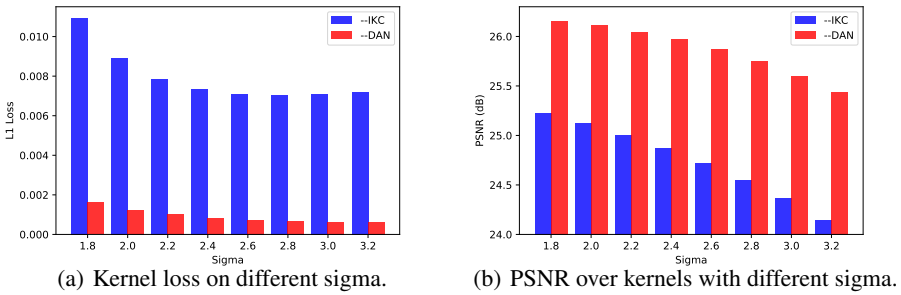
The visual results of *img 892* in DIVKRK are shown in Figure 4. Although the combination of KernelGAN and ZSSR can produce slightly shaper edges than interpolation, it suffers from severe artifacts. The SR image of DAN is obviously much cleaner and has more reliable details.

Table 3: PSNR results when GT kernels are provided.

Methods	Set5	Set14	B100	Urban100	Manga109
DAN	31.89	28.43	27.51	25.86	30.50
DAN(GT kernel)	31.85	28.42	27.51	25.87	30.51

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



(a) Kernel loss on different sigma. (b) PSNR over kernels with different sigma.

Figure 5: The L1 loss of predicted kernels with different sigma (left) and PSNR results with respect to different kernels (right).

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 worser 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 this good cooperation between its *Predictor* and *Restorer*.

4.1.4 Study of Iterations

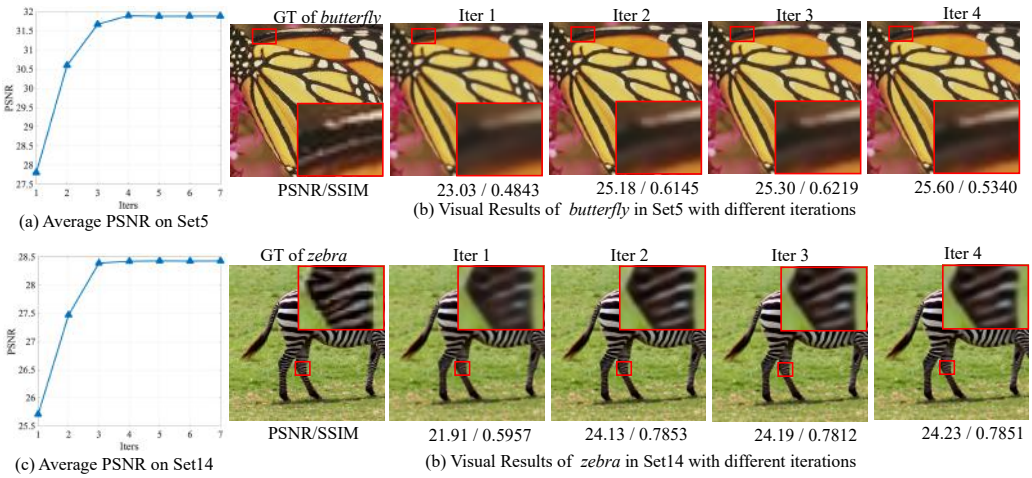
After the model is trained, we also change the number of iterations to see whether the two modules have learned the property of convergence or just have 'remembered' the iteration number. The model is trained with 4 iterations, but during testing we increase the iteration number from 1 to 7. As shown 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 $20.43dB$ when the iteration number is 5, higher than $20.42dB$ 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 *Gaussian8* 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 wold 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.



(a) Average PSNR on Set5

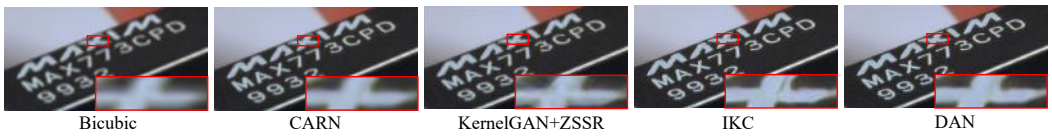
(b) Visual Results of *butterfly* in Set5 with different iterations

(c) Average PSNR on Set14

(b) Visual Results of *zebra* in Set14 with different iterations

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

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.



Bicubic CARN KernelGAN+ZSSR IKC DAN

Figure 7: Visual results on real image *chip*.

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

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1122–1131, 2017.

[2] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 252–268, 2018.

[3] Sefi Bell-Kligler, Assaf Shocher, and Michal Irani. Blind super-resolution kernel estimation using an internal-gan. In *NeurIPS*, 2019.

[4] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie-Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, 2012.

[5] Adrian Bulat, Jing Yang, and Georgios Tzimiropoulos. To learn image super-resolution, use a gan to learn how to do image degradation first. In *ECCV*, 2018.

[6] Victor Cornillere, Abdelaziz Djelouah, Wang Yifan, Olga Sorkine-Hornung, and Christopher Schroers. Blind image super-resolution with spatially variant degradations. *ACM Transactions on Graphics (TOG)*, 38(6):1–13, 2019.

[7] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen O. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16:2080–2095, 2007.

[8] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11065–11074, 2019.

[9] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014.

[10] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *European conference on computer vision*, pages 391–407. Springer, 2016.

[11] Radu Timofte et al. Ntire 2017 challenge on single image super-resolution: Methods and results. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1110–1121, 2017.

[12] Raanan Fattal. Image upsampling via imposed edge statistics. *ACM Trans. Graph.*, 26:95, 2007.

[13] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. *2009 IEEE 12th International Conference on Computer Vision*, pages 349–356, 2009.

[14] Jinjin Gu, Hannan Lu, Wangmeng Zuo, and Chao Dong. Blind super-resolution with iterative kernel correction. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1604–1613, 2019.

[15] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2862–2869, 2014.

[16] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1664–1673, 2018.

[17] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[18] Xuecai Hu, Haoyuan Mu, Xiangyu Zhang, Zilei Wang, Tieniu Tan, and Jian Sun. Meta-sr: A magnification-arbitrary network for super-resolution. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1575–1584, 2019.

[19] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5197–5206, 2015.

[20] Xiaozhong Ji, Yun Cao, Ying Tai, Chengjie Wang, Jilin Li, and Feiyue Huang. Real-world super-resolution via kernel estimation and noise injection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 466–467, 2020.

[21] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017.

[24] David R. Martin, Charless C. Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 2:416–423 vol.2, 2001.

[25] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multimedia Tools and Applications*, 76:21811–21838, 2016.

[26] Tomer Michaeli and Michal Irani. Nonparametric blind super-resolution. *2013 IEEE International Conference on Computer Vision*, pages 945–952, 2013.

[27] Jinshan Pan, Deqing Sun, Hanspeter Pfister, and Ming-Hsuan Yang. Deblurring images via dark channel prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:2315–2328, 2018.

[28] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[29] Assaf Shocher, Nadav Cohen, and Michal Irani. "zero-shot" super-resolution using deep internal learning. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[30] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yi-Hao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks. *ArXiv*, abs/1809.00219, 2018.

[31] Wang Xintao, Yu Ke, Hui Tak-Wa, Dong Chao, Lin Liang, and Change Loy Chen. Deep poly-dense network for image superresolution. 2018.

[32] Yu-Syuan Xu, Shou-Yao Roy Tseng, Yu Hung Tseng, Hsien-Kai Kuo, and Yi-Min Tsai. Unified dynamic convolutional network for super-resolution with variational degradations. *ArXiv*, abs/2004.06965, 2020.

[33] Jiahui Yu, Yuchen Fan, Jianchao Yang, Ning Xu, Zhaowen Wang, Xinchao Wang, and Thomas S. Huang. Wide activation for efficient and accurate image super-resolution. *ArXiv*, abs/1808.08718, 2018.

[34] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, 2010.

[35] Kai Zhang, Luc Van Gool, and Radu Timofte. Deep unfolding network for image super-resolution. *ArXiv*, abs/2003.10428, 2020.

[36] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep cnn denoiser prior for image restoration. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2808–2817, 2017.

[37] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Learning a single convolutional super-resolution network for multiple degradations. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3262–3271, 2018.

[38] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Deep plug-and-play super-resolution for arbitrary blur kernels. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1671–1681, 2019.

[39] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 286–301, 2018.

[40] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2472–2481, 2018.