"KXNet: A Model-Driven Deep Neural Network for Blind Super-Resolution": Supplementary Material

Jiahong Fu[1], Hong Wang[2], Qi Xie*[1], Qian Zhao[1], Deyu Meng[1], and Zongben Xu[1]

[1] Xi'an Jiaotong University, Shaanxi, P.R. China
jiahongfu@stu.xjtu.edu.cn, {xie.qi, timmy.zhaoqian, dymeng, zbxu}@mail.xjtu.edu.cn
[2] Tencent Jarvis Lab, Shenzhen, P.R. China
hazelhwang@tencent.com
[3] Pazhou Lab, Guangzhou, P.R. China

Abstract. In this supplementary material, we provide more details about the optimization algorithm, network module design, and execute more ablation experiments to illustrate the effectiveness of our method. Furthermore, we compare with unsupervised blind super-resolution for blur kernel estimation.

1 Details of Model Optimization

In this section, we provide a detailed derivation in Section 3.2 of the main text. As we shown in the main text, the optimization problem for blind super-resolution can be mathematically expressed as:

$$\begin{aligned} &\min_{\boldsymbol{K},\boldsymbol{X}} \left\| \boldsymbol{Y} - (\boldsymbol{X} \otimes \boldsymbol{K}) \downarrow_{\mathbf{s}} \right\|_F^2 + \lambda_1 \phi_1(\boldsymbol{K}) + \lambda_2 \phi_2(\boldsymbol{X}) \\ &s.t.\ \boldsymbol{K}_j \geq 0, \sum_j \boldsymbol{K}_j = 1, \forall j, \end{aligned} \tag{1}$$

where we aim to estimate a blur kernel $\boldsymbol{K} \in \mathbb{R}^{p \times p}$ and an HR image $\boldsymbol{X} \in \mathbb{R}^{H \times W}$ from an observed LR image $\boldsymbol{Y} \in \mathbb{R}^{h \times w}$; $\phi_1(\boldsymbol{K})$ and $\phi_2(\boldsymbol{X})$ represent the regularization terms for delivering the prior knowledge of blur kernel and HR image, respectively. λ_1 and λ_2 are trade-off regularization parameters. We also introduce the non-negative and equality constraints for every element $\boldsymbol{K}_j$ of blur kernel $\boldsymbol{K}$ to alleviate the non uniqueness of the solution.

As mentioned in Section 3.2 of the main text, we use the proximal gradient algorithm [1] to solve the alternate optimization problem of $\boldsymbol{X}$ and $\boldsymbol{K}$. The details are provided as follows:

Updating blur kernel $\boldsymbol{K}$: The blur kernel $\boldsymbol{K}$ can be updated by solving:

$$\boldsymbol{K}^{(t)} = \arg\min_{\boldsymbol{K}} Q_1\left(\boldsymbol{K}, \boldsymbol{K}^{(t-1)}\right), \tag{2}$$

* Corresponding author.

where $\boldsymbol{K}^{(t-1)}$ denotes the updating result after the last iteration, and $Q_1(\boldsymbol{K}, \boldsymbol{K}^{(t-1)})$ is a quadratic approximation of the objective function Eq. (1) with respect to $\boldsymbol{K}$, mathematically expressed as:

$$
\begin{aligned}
Q_1\left(\boldsymbol{K}, \boldsymbol{K}^{(t-1)}\right) =& f\left(\boldsymbol{K}^{(t-1)}\right) + \frac{1}{2\delta_1}\left\|\boldsymbol{K} - \boldsymbol{K}^{(t-1)}\right\|_F^2 \\
&+ \left\langle \boldsymbol{K} - \boldsymbol{K}^{(t-1)}, \nabla f\left(\boldsymbol{K}^{(t-1)}\right) \right\rangle + \lambda_1\phi_1(\boldsymbol{K}),
\end{aligned} \tag{3}
$$

where $f(\boldsymbol{K}^{(t-1)}) = \left\|\boldsymbol{Y} - \left(\boldsymbol{X}^{(t-1)} \otimes \boldsymbol{K}^{(t-1)}\right) \downarrow_{\mathbf{s}} \right\|_F^2$ and δ_1 denotes the stepsize parameter. Then Eq. (2) is equivalent to:

$$
\begin{aligned}
&\min_{\boldsymbol{K}} \left\|\boldsymbol{K} - \left(\boldsymbol{K}^{(t-1)} - \delta_1 \nabla f\left(\boldsymbol{K}^{(t-1)}\right)\right)\right\|_F^2 + \lambda_1\delta_1\phi_1(\boldsymbol{K}) \\
&s.t.\ \boldsymbol{K}_j \geq 0, \sum_j \boldsymbol{K}_j = 1, \forall j,
\end{aligned} \tag{4}
$$

It's solution can then be easily expressed in close-form as [4]:

$$
\boldsymbol{K}^{(t)} = \operatorname{prox}_{\lambda_1\delta_1}\left(\boldsymbol{K}^{(t-1)} - \delta_1 \nabla f\left(\boldsymbol{K}^{(t-1)}\right)\right), \tag{5}
$$

where $\operatorname{prox}_{\lambda_1\delta_1}(\cdot)$ is the proximal operator dependent on the regularization term $\phi_1(\cdot)$ with respect to $\boldsymbol{K}$; the specific form of $\nabla f(\boldsymbol{K}^{(t-1)})$ is complicated. For ease of calculation by transforming the convolutional operation in $f(\boldsymbol{K}^{(t-1)})$ into matrix multiplication, as shown in the main text, we have:

$$
\nabla f\left(\boldsymbol{k}^{(t-1)}\right) = \left(D_{\mathbf{s}} U_f\left(\boldsymbol{X}^{(t-1)}\right)\right)^{\mathrm{T}} \operatorname{vec}\left(\boldsymbol{Y} - \left(\boldsymbol{X}^{(t-1)} \otimes \boldsymbol{K}^{(t-1)}\right)\downarrow_{\mathbf{s}}\right), \tag{6}
$$

where $U_f\left(\boldsymbol{X}^{(t-1)}\right) \in R^{HW\times p^2}$ are the unfolded result of $\boldsymbol{X}^{(t-1)}$; $D_{\mathbf{s}}$ denotes the downsampling operator which is corresponding to the operator $\downarrow_{\mathbf{s}}$, and achieves the transformation from the size HW to the size hw. Thus, the result $D_{\mathbf{s}}U_f\left(\boldsymbol{X}^{(t-1)}\right)$; $\nabla f\left(\boldsymbol{k}^{(t-1)}\right) \in \mathbb{R}^{p^2\times 1}$; $\nabla f\left(\boldsymbol{K}^{(t-1)}\right) = \operatorname{vec}^{-1}\left(\nabla f(\boldsymbol{k}^{(t-1)})\right)$; $\operatorname{vec}^{-1}(\cdot)$ is the reverse vectorization;

Implementation of $D_{\mathbf{s}}U_f(\cdot)$: With Pytorch [4] framework, we can directly perform "torch.nn.function.unfold" with $stride = \mathbf{s}$ on $\boldsymbol{X}^{(t-1)} \in \mathbb{R}^{H\times W}$ to get $\left(D_{\mathbf{s}}U_f(\boldsymbol{X}^{(t-1)})\right)^{\mathrm{T}} \in \mathbb{R}^{p^2\times hw}$, and execute "torch.permute" to get $D_{\mathbf{s}}U_f(\boldsymbol{X}^{(t-1)}) \in \mathbb{R}^{hw\times p^2}$.

Updating HR image $\boldsymbol{X}$: Similarly, the quadratic approximation of the problem in Eq. (1) with respect to $\boldsymbol{X}$ is:

$$
\begin{aligned}
Q_2\left(\boldsymbol{X}, \boldsymbol{X}^{(t-1)}\right) =& h\left(\boldsymbol{X}^{(t-1)}\right) + \frac{1}{2\delta_2}\left\|\boldsymbol{X} - \boldsymbol{X}^{(t-1)}\right\|_F^2 \\
&+ \left\langle \boldsymbol{X} - \boldsymbol{X}^{(t-1)}, \nabla h\left(\boldsymbol{X}^{(t-1)}\right) \right\rangle + \lambda_2\phi_2(\boldsymbol{X}),
\end{aligned} \tag{7}
$$

[4] https://pytorch.org/

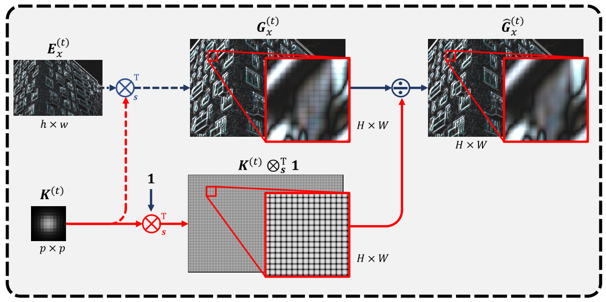

Fig. 1. Illustration of the gradient adjuster. The solid line represents the gradient adjustment process.

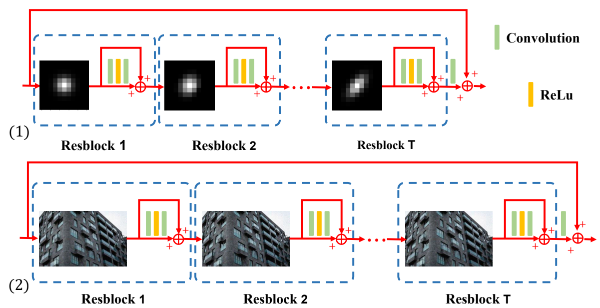

Fig. 2. (1) The exploited ResNet for the proximal network $\mathrm{proxNet}_{\theta_k^{(t)}}(\cdot)$. (2) The exploited ResNet for the proximal network $\mathrm{proxNet}_{\theta_x^{(t)}}(\cdot)$

where $h(\boldsymbol{X}^{(t-1)}) = \left\|\boldsymbol{Y} - \left(\boldsymbol{X}^{(t-1)} \otimes \boldsymbol{K}^{(t)}\right) \downarrow_{\mathbf{s}}\right\|_F^2$; $\nabla h\left(\boldsymbol{X}^{(t-1)}\right) = \boldsymbol{K}^{(t)} \otimes_{\mathbf{s}}^{\mathrm{T}} \left(\boldsymbol{Y} - \left(\boldsymbol{X}^{(t-1)} \otimes \boldsymbol{K}^{(t)}\right) \downarrow_{\mathbf{s}}\right)$; δ_2 denotes the stepsize parameter. Then the equivalent optimization problem is:

$$\min_{\boldsymbol{X}} \left\|\boldsymbol{X} - \left(\boldsymbol{X}^{(t-1)} - \delta_2 \nabla h\left(\boldsymbol{X}^{(t-1)}\right)\right)\right\|_F^2 + \lambda_2 \delta_2 \phi_2(\boldsymbol{X}), \tag{8}$$

Similarly, we can easily deduce the updating rule for X as:

$$\boldsymbol{X}^{(t)} = \mathrm{prox}_{\lambda_2 \delta_2}\Big(\boldsymbol{X}^{(t-1)} - \delta_2 \boldsymbol{K}^{(t)} \otimes_{\mathbf{s}}^{\mathrm{T}} \left(\boldsymbol{Y} - (\boldsymbol{X}^{(t-1)} \otimes \boldsymbol{K}^{(t)}) \downarrow_{\mathbf{s}}\right)\Big), \tag{9}$$

where $\mathrm{prox}_{\lambda_2 \delta_2}(\cdot)$ is the proximal operator dependent on the regularization term $\phi_2(\cdot)$ with respect to $\boldsymbol{X}$.

2 Details of Network Module Design

In this section, we provide more details of network design, including the gradient adjuster, $\mathrm{proxNet}_{\theta_k^{(t)}}(\cdot)$ and $\mathrm{proxNet}_{\theta_x^{(t)}}(\cdot)$.

Gradient adjuster. As stated in Section 4.1 of the main text, we adopt an adjuster to the gradient $\boldsymbol{G}_x^{(t)}$, which alleviate the unevenness issue. As shown in

Table 1. Average PSNR/SSIM of adopting different strategies for X-net on synthesized testing sets. KXNet* presents without gradient adjuster and concatenating strategy.

Method	Noise	Urban100 [7]		BSD100 [10]		Set14 [14]		Set5 [3]	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
KXNet*	0	27.55	0.8425	29.71	0.8293	30.45	0.8532	33.63	0.9213
KXNet(ours)		**28.33**	**0.8627**	**30.21**	**0.8456**	**31.14**	**0.8672**	**34.59**	**0.9315**
KXNet*	5	26.51	0.7957	28.21	0.7580	29.01	0.7951	31.82	0.8829
KXNet(ours)		**26.88**	**0.8056**	**28.33**	**0.7615**	**29.22**	**0.7993**	**32.07**	**0.8864**
KXNet*	15	25.25	0.7433	26.82	0.6946	27.53	0.7402	29.84	0.8422
KXNet(ours)		**25.45**	**0.7500**	**26.87**	**0.6959**	**27.59**	**0.7422**	**29.93**	**0.8449**

Fig. 1, the residual image $\boldsymbol{E}_x^{(t)}$ are deconvolved with blur kernel $\boldsymbol{K}^{(t)}$ to obtain the gradient $\boldsymbol{G}_x^{(t)}$. We can clearly find that due to the "uneven overlap" phenomenon with transposed convolution, the obtained gradient $\boldsymbol{G}_x^{(t)}$ is corrupted with unexpected grid-like artifacts. These kinds of artifacts can be detrimental to image restoration for it is neither smooth nor natural. To alleviate the unevenness issue, we introduce the $\boldsymbol{K}^{(t)} \otimes_{\mathbf{s}}^{\mathrm{T}} \mathbf{1}$ to calculate the degree of uneven overlap and element-wisely divide $\boldsymbol{G}_x^{(t)}$ with $\boldsymbol{K}^{(t)} \otimes_{\mathbf{s}}^{\mathrm{T}} \mathbf{1}$ to get an adjusted gradient, $\hat{\boldsymbol{G}}_x^{(t)}$. As illustrated in Fig. 1, $\hat{\boldsymbol{G}}_x^{(t)}$ has more precise textures and edges than $\boldsymbol{G}_x^{(t)}$, which will improve the recovery performance of X-net.

Proximal Network Architecture. As stated in the main text, the proximal network $\mathrm{proxNet}_{\theta_k^{(t)}}(\cdot)$ and $\mathrm{proxNet}_{\theta_x^{(t)}}(\cdot)$ $(t = 0, 1, \cdots, T)$ are two shallow ResNets. Fig. 2 shows the architectural details of $\mathrm{proxNet}_{\theta_k^{(t)}}(\cdot)$ and $\mathrm{proxNet}_{\theta_x^{(t)}}(\cdot)$, respectively. For each Resblock in each stage in $\mathrm{proxNet}_{\theta_k^{(t)}}(\cdot)$, we simply adopt the same structure. $\mathrm{proxNet}_{\theta_x^{(t)}}(\cdot)$ also uses the same strategy. It is worth noting that we also adopted the residual in residual (RIR) structure [16] here, which is very effective for single image super-resolution problems.

3 Ablation studies

In this section, we will provide the ablation studies about the gradient adjuster and concatenating $\boldsymbol{X}^{(t-1)}$ and $\hat{\boldsymbol{G}}_x^{(t)}$ as the input of the proximal network $\mathrm{proxNet}_{\theta_x^{(t)}}(\cdot)$. In this part, the setting of the experiment is the same as setting 2 for scale factor 2 in the main text. We adopt the PSNR and SSIM computed on Y channel in the YCbCr space for quantitative analysis. As shown in Table 1, if we do not perform gradient adjuster and concatenate $\boldsymbol{X}^{(t-1)}$ and $\hat{\boldsymbol{G}}_x^{(t)}$ as the input of the proximal network $\mathrm{proxNet}_{\theta_x^{(t)}}(\cdot)$, then the performance will be greatly reduced. The above experiments verify the effectiveness of gradient adjuster and concatenation strategy.

Table 2. Compared to USRNet on Set14.

Method (x2)	K-Net+USRNet [15]	KXNet
PSNR / SSIM	29.37 / 0.8250	29.71 / 0.8354
Speed (seconds)	1.38	0.47

4 Compare with the Non-Blind Super-Resolution Unfolding Method.

Here we mainly compare with the SOTA of the non-blind super-resolution unfolding method, USRNet[15]. Firstly, our method targets the blind super-resolution problem, i.e., the blur kernel is unknown, while USRNet targets the non-blind super-resolution problem. Therefore, USRNet has no function of estimating the blur kernel and handling the blur kernel unknown cases. Secondly, compared with USRNet, the X-Net we constructed has the following advantages: **1) Simpler operators.** When updating $X^{(t)}$, USRNet involves multiple Fourier transforms and division operations, which will increase the computational complexity, especially during the gradient backpropagation process. Comparatively, KXNet only contains convolutions and ReLu operations, which are evidently simpler to calculate. Thus, the inference speed of KXNet is much faster, as illustrated in Table 2. **2) More stable results.** KXNet is built according to a proximal gradient descent algorithm, and the updating of X and K are both performed by gradient descent and ResNet, which adjusts X and K by adding a relatively small residue. This updating manner is very stable, ensuring that X and K don't change very much through network stages. In comparison, the manner for updating X in USRNet could always be hardly guaranteed to be stable, for the Fourier transforms and inverse Fourier transforms tend to make the updating results more unpredictable. To verify this point, we replace our X-Net with USRNet, i.e., combining K-Net with USRNet (denoted K-Net+USRNet), and conduct experiments under the same setting with KXNet. The experimental results are shown in the following Table 2. We controlled the number of stages of KXNet so that the parameters of the two methods are almost the same, and the advantage of KXNet can be evidently observed.

Table 3. Performance comparison between SeaNet/ENLCA and KXNet on Set14.

Method (x2)	SeaNet [5]	ENLCA [13]	KXNet
PSNR	29.50	18.35	31.14
SSIM	0.8262	0.5077	0.8672

5 Compare with other Super-Resolution Method.

We further compare with the latest super-resolution algorithms, SeaNet [5] and ENLCA [13]. Thus, we conduct experiments under the same setting with KXNet, and the final results are shown in Table 3.

Table 4. Average PSNR/SSIM of all the comparing methods **(Setting 2)**.

Method	Scale	Urban100 [7]		BSD100 [10]		Set14 [14]		Set5 [3]		Noise
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	Level
Bicubic		23.00	0.6656	25.85	0.6769	25.74	0.7085	27.68	0.8047	
RCAN [16]		23.22	0.6791	26.03	0.6896	25.92	0.7217	27.85	0.8095	
IKC [6]	x2	27.46	0.8401	29.85	0.8390	30.69	0.8614	33.99	0.9229	
DASR [12]		26.65	0.8106	28.84	0.7965	29.44	0.8224	32.50	0.8961	
DAN [9]		27.93	0.8497	30.09	0.8410	31.03	0.8647	34.40	0.9291	
KXNet(ours)		**28.33**	**0.8627**	**30.21**	**0.8456**	**31.14**	**0.8672**	**34.59**	**0.9315**	
Bicubic		21.80	0.6084	24.68	0.6254	24.28	0.6546	25.78	0.7555	
RCAN [16]		21.38	0.6042	24.47	0.6299	24.07	0.6606	25.63	0.7572	
IKC [6]	x3	25.36	0.7626	27.56	0.7475	28.19	0.7805	31.60	0.8853	
DASR [12]		25.20	0.7575	27.39	0.7379	27.96	0.7727	30.91	0.8723	0
DAN [9]		25.82	0.7855	27.88	0.7603	28.69	0.7969	31.70	0.8940	
KXNet(ours)		**26.37**	**0.8035**	**28.15**	**0.7672**	**29.04**	**0.8036**	**32.53**	**0.9034**	
Bicubic		20.88	0.5602	23.75	0.5827	23.17	0.6082	24.35	0.7086	
RCAN [16]		19.84	0.5307	23.10	0.5729	22.38	0.5967	23.72	0.6973	
IKC [6]	x4	24.33	0.7241	26.49	0.6968	27.04	0.7398	29.60	0.8503	
DASR [12]		24.20	0.7150	26.43	0.6903	26.89	0.7306	29.53	0.8455	
DAN [9]		24.91	0.7491	26.92	0.7168	27.69	0.7600	30.53	0.8746	
KXNet(ours)		**25.30**	**0.7647**	**27.08**	**0.7221**	**27.98**	**0.7659**	**30.99**	**0.8815**	
Bicubic		22.89	0.6401	25.65	0.6498	25.55	0.6826	27.39	0.7738	
RCAN [16]		22.88	0.5986	25.43	0.6092	25.34	0.6460	26.89	0.7333	
IKC [6]	x2	25.72	0.7678	27.76	0.7456	28.28	0.7793	31.23	0.8718	
DASR [12]		25.89	0.7739	27.75	0.7396	28.41	0.7767	31.03	0.8680	
DAN [9]		26.64	0.7964	28.24	0.7571	29.14	0.7954	32.01	0.8843	
KXNet(ours)		**26.88**	**0.8056**	**28.33**	**0.7615**	**29.22**	**0.7993**	**32.07**	**0.8864**	
Bicubic		21.72	0.5889	24.52	0.6045	24.15	0.6349	25.59	0.7316	
RCAN [16]		21.30	0.5538	24.19	0.5776	23.80	0.6099	25.23	0.7128	
IKC [6]	x3	25.01	0.7405	26.95	0.7070	27.59	0.7481	30.53	0.8586	
DASR [12]		24.73	0.7300	26.75	0.6957	27.32	0.7379	29.84	0.8437	5
DAN [9]		25.13	0.7472	27.01	0.7090	27.72	0.7513	30.36	0.8578	
KXNet(ours)		**25.49**	**0.7614**	**27.13**	**0.7140**	**27.92**	**0.7572**	**30.71**	**0.8637**	
Bicubic		20.82	0.5461	23.63	0.5676	23.07	0.5937	24.22	0.6911	
RCAN [16]		19.92	0.4996	22.99	0.5365	22.31	0.5597	23.48	0.6633	
IKC [6]	x4	24.02	0.7027	26.03	0.6664	26.57	0.7129	28.90	0.8275	
DASR [12]		23.92	0.6982	26.01	0.6636	26.46	0.7068	28.69	0.8211	
DAN [9]		24.33	0.7167	26.20	0.6752	26.86	0.7219	29.30	0.8404	
KXNet(ours)		**24.59**	**0.7285**	**26.28**	**0.6777**	**27.01**	**0.7262**	**29.55**	**0.8450**	
Bicubic		22.19	0.5159	24.44	0.5150	24.38	0.5497	25.72	0.6241	
RCAN [16]		21.28	0.3884	22.98	0.3822	22.96	0.4155	23.76	0.4706	
IKC [6]	x2	24.69	0.7208	26.49	0.6828	26.93	0.7244	29.21	0.8260	
DASR [12]		24.84	0.7273	26.63	0.6841	27.22	0.7283	29.44	0.8322	
DAN [9]		25.32	0.7447	26.84	0.6932	27.56	0.7392	29.91	0.8430	
KXNet(ours)		**25.45**	**0.7500**	**26.87**	**0.6959**	**27.59**	**0.7422**	**29.93**	**0.8449**	
Bicubic		21.18	0.4891	23.55	0.4961	23.28	0.5289	24.42	0.6119	
RCAN [16]		20.22	0.3693	22.20	0.3726	21.99	0.4053	22.85	0.4745	
IKC [6]	x3	24.21	0.7019	25.93	0.6564	26.42	0.7018	28.61	0.8135	
DASR [12]		23.93	0.6890	25.82	0.6484	26.27	0.6940	28.27	0.8047	15
DAN [9]		24.17	0.7013	25.93	0.6551	26.46	0.7014	28.52	0.8130	
KXNet(ours)		**24.42**	**0.7135**	**25.99**	**0.6585**	**26.56**	**0.7063**	**28.64**	**0.8178**	
Bicubic		20.38	0.4690	22.83	0.4841	22.39	0.5120	23.33	0.5977	
RCAN [16]		19.23	0.3515	21.47	0.3686	21.05	0.3960	21.77	0.4689	
IKC [6]	x4	23.35	0.6665	25.21	0.6238	25.58	0.6712	27.45	0.7867	
DASR [12]		23.26	0.6620	25.20	0.6223	25.55	0.6683	27.32	0.7842	
DAN [9]		23.48	0.6742	25.25	0.6283	25.72	0.6760	27.55	0.7938	
KXNet(ours)		**23.67**	**0.6844**	**25.30**	**0.6296**	**25.78**	**0.6792**	**27.66**	**0.7977**	

6 More Experimental Results

In this section, we provide more numerical results of KXNet on Setting2 in the main text, as shown in Table 4.

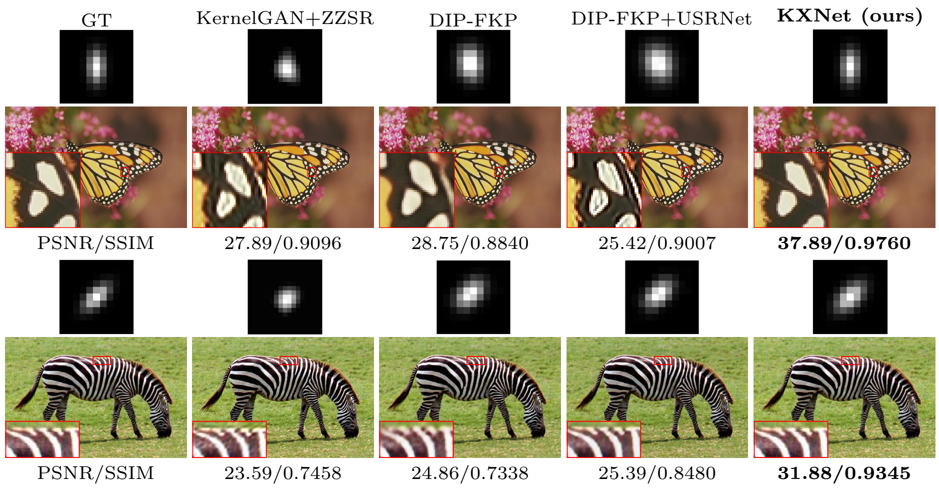

Fig. 3. Blur kernel estimation and performance comparison on *img 11* and *img 14* in Set14 [14]. The scale factor is 2 and the noise level is 0. The blur kernel estimated by DIP-FKP shifts to the upper left due to the underlying assumption.

Table 5. Quantitative comparison with the unsupervised SOTA methods.

Noise Level	Method (x2)	Set14 [14] PSNR	Set14 [14] SSIM
0	KernelGAN [2] + ZSSR [11]	24.88	0.7532
	DIP-FKP [8]	25.81	0.7210
	DIP-FKP [8] + USRNet [15]	22.58	0.7317
	KXNet(ours)	**32.10**	**0.8979**

7 Blur Kernel Estimation

In this section, we demonstrate more experimental results about blur kernel estimation. Currently, there are some unsupervised blind super-resolution methods that have achieved remarkable performance in estimating blur kernel, such as KernelGAN [2] and DIP-FKP [8]. We can combine the estimated blur kernels by these methods and the competing non-blind SR methods, such as ZZSR and USRNet, to accomplish the blind SR task. As shown in Fig. 3 and Table 5, due to the single image learning strategy, these unsupervised methods cannot learn rich image priors underlying data and the SR performance is inferior. Besides, by comparing DIP-FKP and DIP-FKP+USRNet, we can easily find that the inaccuracy of the estimated blur kernel by DIP-FKP tends to adversely affect the SR performance of the non-blind SR method–USRNet. In contrast, under different blur kernel degradation settings, the proposed method can consistently achieve better kernel estimation and obtain better SR performance, which significantly outperforms the KernelGAN+ZSSR and DIP-FKP+USRNet. This finely substantiates the superiority of our unfolding network which fully and reasonably embeds the inherent relationship between blur kernel and HR image. It is the joint estimation of blur kernel and HR image which guides the network to learn in the right direction.

References

1. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM journal on imaging sciences **2**(1), 183–202 (2009)
2. Bell-Kligler, S., Shocher, A., Irani, M.: Blind super-resolution kernel estimation using an internal-gan. arXiv preprint arXiv:1909.06581 (2019)
3. Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M.L.: Low-complexity single-image super-resolution based on nonnegative neighbor embedding (2012)
4. Donoho, D.L.: De-noising by soft-thresholding. IEEE transactions on information theory **41**(3), 613–627 (1995)
5. Fang, F., Li, J., Zeng, T.: Soft-edge assisted network for single image super-resolution. IEEE Transactions on Image Processing **29**, 4656–4668 (2020)
6. Gu, J., Lu, H., Zuo, W., Dong, C.: Blind super-resolution with iterative kernel correction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1604–1613 (2019)
7. Huang, J.B., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5197–5206 (2015)
8. Liang, J., Zhang, K., Gu, S., Van Gool, L., Timofte, R.: Flow-based kernel prior with application to blind super-resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10601–10610 (2021)
9. Luo, Z., Huang, Y., Li, S., Wang, L., Tan, T.: Unfolding the alternating optimization for blind super resolution. arXiv preprint arXiv:2010.02631 (2020)
10. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001. vol. 2, pp. 416–423. IEEE (2001)
11. Shocher, A., Cohen, N., Irani, M.: "zero-shot" super-resolution using deep internal learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3118–3126 (2018)
12. Wang, L., Wang, Y., Dong, X., Xu, Q., Yang, J., An, W., Guo, Y.: Unsupervised degradation representation learning for blind super-resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10581–10590 (2021)
13. Xia, B., Hang, Y., Tian, Y., Yang, W., Liao, Q., Zhou, J.: Efficient non-local contrastive attention for image super-resolution (2022)
14. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: International conference on curves and surfaces. pp. 711–730. Springer (2010)
15. Zhang, K., Gool, L.V., Timofte, R.: Deep unfolding network for image super-resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3217–3226 (2020)
16. Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., Fu, Y.: Image super-resolution using very deep residual channel attention networks. In: Proceedings of the European conference on computer vision (ECCV). pp. 286–301 (2018)