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

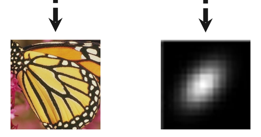
Jiahong Fu[1], Hong Wang[2], Qi Xie*[1], Qian Zhao[1], Deyu Meng[1,3], Zongben Xu[1,3]

[1]Xi'an Jiaotong University; [2]Tencent Jarvis Lab, Shenzhen; [3]Pazhou Lab, Guangzhou

ECCV TEL AVIV 2022

Blind Super-Resolution

$$\boldsymbol{Y} = (\boldsymbol{X} \otimes \boldsymbol{K}) \downarrow_s + \boldsymbol{N}$$



- Estimat **blur kernel** $\boldsymbol{K}$
- Estimat **high resolution image** $\boldsymbol{X}$

Contributions

- We propose a novel model-driven deep unfolding blind super-resolution network, named KXNet.
- The proposed scheme can explicitly estimate the blur kernel with clear physical structures.
- With intrinsic embedding of the physical generation mechanism, we maintain the essential convolution computation between blur kernel and HR image.

Model Formulation

$$\min_{\boldsymbol{K},\boldsymbol{X}} \|\boldsymbol{Y} - (\boldsymbol{X}\otimes\boldsymbol{K}) \downarrow_s\|_F^2 + \lambda_1\phi_1(\boldsymbol{K}) + \lambda_2\phi_2(\boldsymbol{X})$$

$$s.t.\ \boldsymbol{K}_j \geq 0, \sum_j \boldsymbol{K}_j = 1, \forall j,$$

Model Optimization

For blur kernel $\boldsymbol{K}$:

$$\min_{\boldsymbol{K}} \|\boldsymbol{Y} - (\boldsymbol{X}\otimes\boldsymbol{K}) \downarrow_s\|_F^2 + \lambda_1\phi_1(\boldsymbol{K})$$

$$s.t.\ \boldsymbol{K}_j \geq 0, \sum_j \boldsymbol{K}_j = 1, \forall j,$$

Solving with proximal gradient algorithm with iteration:

$$\boldsymbol{K}^{(t)} = \mathrm{prox}_{\lambda_1\delta_1}(\boldsymbol{K}^{(t-1)} - \delta_1 \nabla f(\boldsymbol{K}^{(t-1)}))$$

where $\nabla f\left(\boldsymbol{K}^{(t-1)}\right) = \mathrm{vec}^{-1}(\nabla f(\boldsymbol{k}^{(t-1)}))$, and

$$\nabla f\left(\boldsymbol{k}^{(t-1)}\right) = \left(D_s U_f\left(\boldsymbol{X}^{(t-1)}\right)\right)^{\mathrm{T}} \mathrm{vec}\left(\boldsymbol{Y} - \left(\boldsymbol{X}^{(t-1)}\otimes\boldsymbol{K}^{(t-1)}\right)\downarrow_s\right)$$

For image $\boldsymbol{X}$:

$$\min_{\boldsymbol{X}} \|\boldsymbol{Y} - (\boldsymbol{X}\otimes\boldsymbol{K}) \downarrow_s\|_F^2 + \lambda_2\phi_2(\boldsymbol{X})$$

Same as above:

$$\boldsymbol{X}^{(t)} = \mathrm{prox}_{\lambda_2\delta_2}(\boldsymbol{X}^{(t-1)} - \delta_2 \boldsymbol{K}^{(t)} \otimes_s^{\mathrm{T}} (\boldsymbol{Y} - \left(\boldsymbol{X}^{(t-1)}\otimes\boldsymbol{K}^{(t)}\right)\downarrow_s))$$

Network Module

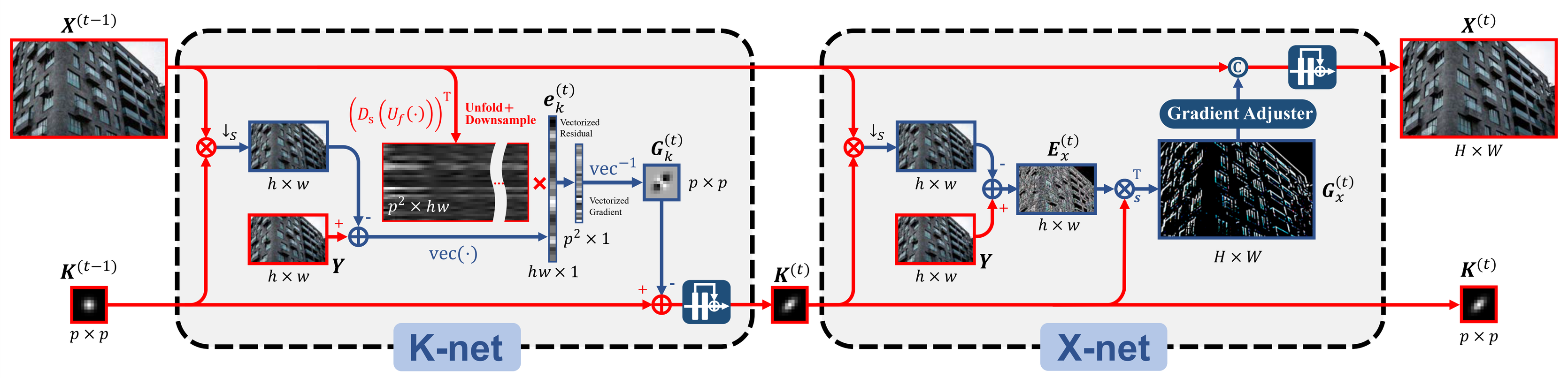
$\boldsymbol{K}$ − net:

Iterative optimization algorithm	Network design
For $t = 1{:}T$ do:	In stage $t = 1{:}T$ of the network do:
$\varepsilon_k^{(t)} = \mathrm{vec}(\boldsymbol{Y} - (\boldsymbol{X}^{(t-1)}\otimes\boldsymbol{K}^{(t-1)})\downarrow_s)$	$e_k^{(t)} = \mathrm{vec}(\boldsymbol{Y} - (\boldsymbol{X}^{(t-1)}\otimes\boldsymbol{K}^{(t-1)})\downarrow_s)$
$\mathcal{G}_k^{(t)} = \mathrm{vec}^{-1}\left(\left(D_s U_f(\boldsymbol{X}^{(t-1)})\right)^{\mathrm{T}} \varepsilon_k^{(t)}\right)$	$\boldsymbol{G}_k^{(t)} = \mathrm{vec}^{-1}\left(\left(D_s U_f(\boldsymbol{X}^{(t-1)})\right)^{\mathrm{T}} e_k^{(t)}\right)$
$\boldsymbol{K}^{(t)} = \mathrm{prox}_{\lambda_1\delta_1}(\boldsymbol{K}^{(t-1)} - \delta_1\mathcal{G}_k^{(t)})$	$\boldsymbol{K}^{(t)} = \mathrm{prox}_{\theta_k^{(t)}}(\boldsymbol{K}^{(t-1)} - \delta_1\boldsymbol{G}_k^{(t)})$

$\boldsymbol{X}$ − net:

Iterative optimization algorithm	Network design
For $t = 1{:}T$ do:	In stage $t = 1{:}T$ of the network do:
$\mathcal{E}_x^{(t)} = \boldsymbol{Y} - (\boldsymbol{X}^{(t-1)}\otimes\boldsymbol{K}^{(t)})\downarrow_s$	$\boldsymbol{E}_x^{(t)} = \boldsymbol{Y} - (\boldsymbol{X}^{(t-1)}\otimes\boldsymbol{K}^{(t)})\downarrow_s$
$\mathcal{G}_x^{(t)} = \boldsymbol{K}^{(t)} \otimes_s^{\mathrm{T}} \mathcal{E}_x^{(t)}$	$\boldsymbol{G}_x^{(t)} = \boldsymbol{K}^{(t)} \otimes_s^{\mathrm{T}} \boldsymbol{E}_x^{(t)}$
$\widehat{\mathcal{G}}_x^{(t)} = \mathrm{adjuster}(\mathcal{G}_x^{(t)})$	$\widehat{\boldsymbol{G}}_k^{(t)} = \mathrm{adjuster}(\boldsymbol{G}_x^{(t)})$
$\boldsymbol{X}^{(t)} = \mathrm{prox}_{\lambda_2\delta_2}(\boldsymbol{X}^{(t-1)} - \delta_1\widehat{\mathcal{G}}_x^{(t)})$	$\boldsymbol{X}^{(t)} = \mathrm{prox}_{\theta_x^{(t)}}(\boldsymbol{X}^{(t-1)}, \widehat{\boldsymbol{G}}_k^{(t)})$

Model-Driven Architecture



vec(·) : vectorization $U_f(\cdot)$: unfolding D_s: downsampling $(\cdot)^{\mathrm{T}}$: matrix transposing : proxNet © : concat

Training Loss: $$L = \sum_{t=1}^{T} \alpha_t \left\|\boldsymbol{K} - \boldsymbol{K}^{(t)}\right\|_1 + \sum_{t=1}^{T} \beta_t \left\|\boldsymbol{X} - \boldsymbol{X}^{(t)}\right\|_1$$

Experimental Results

Table 1. Averaged PSNR/SSIM results of the comparison methods

Method	Scale	Urban100 [17] PSNR	Urban100 [17] SSIM	BSD100 [28] PSNR	BSD100 [28] SSIM	Set14 [52] PSNR	Set14 [52] SSIM	Set5 [4] PSNR	Set5 [4] SSIM	Noise Level
Bicubic	x2	23.00	0.6656	25.85	0.6769	25.74	0.7085	27.68	0.8047	0
RCAN [59]		23.22	0.6791	26.03	0.6896	25.92	0.7217	27.85	0.8095	
IKC [14]		27.46	0.8401	29.85	0.8390	30.69	0.8614	33.99	0.9229	
DASR [43]		26.65	0.8106	28.84	0.7965	29.44	0.8224	32.50	0.8961	
DAN [25]		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	x3	21.80	0.6084	24.68	0.6254	24.28	0.6546	25.78	0.7555	
RCAN [59]		21.38	0.6042	24.47	0.6299	24.07	0.6606	25.63	0.7572	
IKC [14]		25.36	0.7626	27.56	0.7475	28.19	0.7805	31.60	0.8853	
DASR [43]		25.20	0.7575	27.39	0.7379	27.96	0.7727	30.91	0.8723	
DAN [25]		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	x4	20.88	0.5602	23.75	0.5827	23.17	0.6082	24.35	0.7086	
RCAN [59]		19.84	0.5307	23.10	0.5729	22.38	0.5967	23.72	0.6973	
IKC [14]		24.33	0.7241	26.49	0.6968	27.04	0.7398	29.60	0.8503	
DASR [43]		24.20	0.7150	26.43	0.6903	26.89	0.7306	29.53	0.8455	
DAN [25]		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	x2	22.19	0.5159	24.44	0.5150	24.38	0.5497	25.72	0.6241	15
RCAN [59]		21.28	0.3884	22.98	0.3822	22.96	0.4155	23.76	0.4706	
IKC [14]		24.69	0.7208	26.49	0.6828	26.93	0.7244	29.21	0.8260	
DASR [43]		24.84	0.7273	26.63	0.6841	27.22	0.7283	29.44	0.8322	
DAN [25]		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	x3	21.18	0.4891	23.55	0.4961	23.28	0.5289	24.42	0.6119	
RCAN [59]		20.22	0.3693	22.20	0.3726	21.99	0.4053	22.85	0.4745	
IKC [14]		24.21	0.7019	25.93	0.6564	26.42	0.7018	28.61	0.8135	
DASR [43]		23.93	0.6890	25.82	0.6484	26.27	0.6940	28.27	0.8047	
DAN [25]		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**	

Figure 1. Super-resolution results of different methods

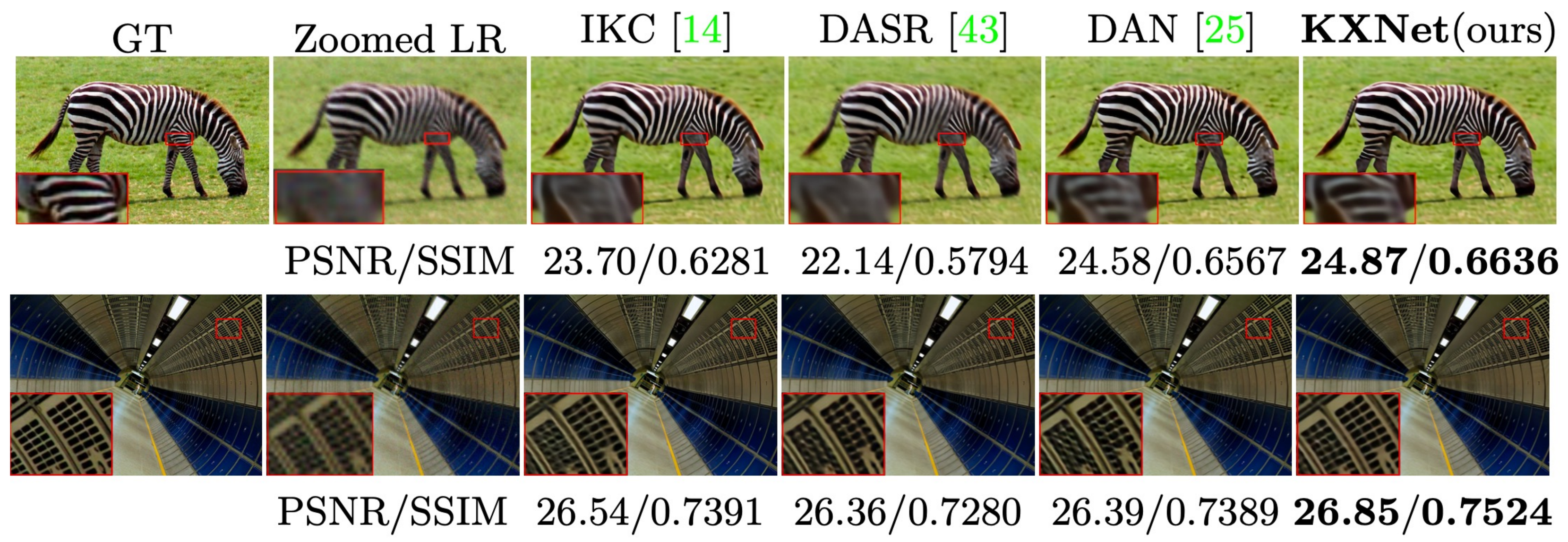
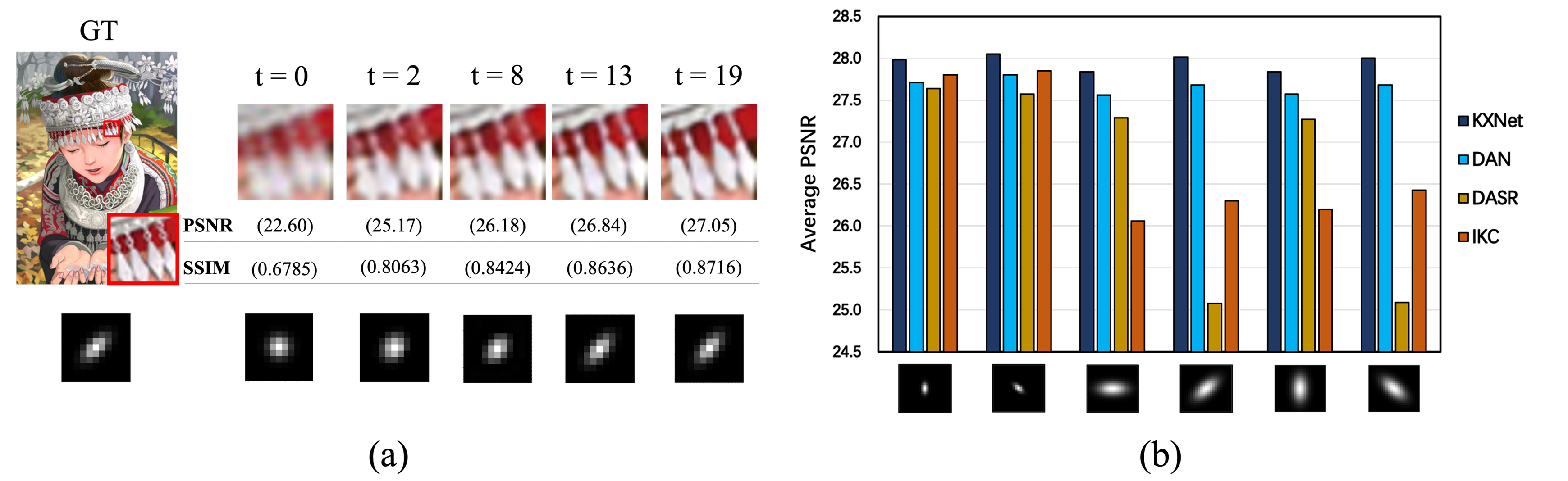


Table 2. Effect of stage number S on the performance of KXNet on Set14.

Stage No.	S=0	S=5	S=10	S=17	S=19	S=21
PSNR	25.74	29.91	30.57	30.96	31.14	31.14
SSIM	0.7085	0.8400	0.8556	0.8631	0.8672	0.8665
Params(M)	-	1.72	3.42	5.82	6.50	7.18
Speed(seconds)	-	0.51	0.54	0.58	0.59	0.64

Figure 2. (a) The estimated SR image and the extracted blur kernel at different iterative stages of KXNet. (b) Performance comparison under different blur kernel setting on Set14 (scale=4, noise=0).



Paper & Codes

https://github.com/jiahong-fu/KXNet