“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:

\[
\min_{K, X} \| Y - (X \otimes K) \downarrow s \|_F + \lambda_1 \phi_1(K) + \lambda_2 \phi_2(X)
\]

subject to \( K_j \geq 0 \), \( K_j^T K_j = 1 \), \( \forall j \).

(1)

where we aim to estimate a blur kernel \( K \in \mathbb{R}^{p \times p} \) and an HR image \( X \in \mathbb{R}^{H \times W} \) from an observed LR image \( Y \in \mathbb{R}^{h \times w} \); \( \phi_1(K) \) and \( \phi_2(X) \) represent the regularization terms for delivering the prior knowledge of blur kernel and HR image, respectively. \( \lambda_1 \) and \( \lambda_2 \) are trade-off regularization parameters. We also introduce the non-negative and equality constraints for every element \( K_j \) of blur kernel \( K \) 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 \( X \) and \( K \). The details are provided as follows:

**Updating blur kernel \( K \):** The blur kernel \( K \) can be updated by solving:

\[
K^{(t)} = \arg \min_K Q_1(K, K^{(t-1)})
\]

(2)

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

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

where $f(K^{(t-1)}) = \|Y - (X^{(t-1)} \otimes K^{(t-1)}) d_s\|_F^2$ and $\delta_1$ denotes the stepsize parameter. Then Eq. (2) is equivalent to:

$$\min_{K} \left\| K - (K^{(t-1)} - \delta_1 \nabla f(K^{(t-1)})) \right\|_F^2 + \lambda_1 \delta_1 \phi_1(K) \quad \text{s.t. } K_j \geq 0, \sum_j K_j = 1, \forall j,$$

(4)

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

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

(5)

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

$$\nabla f(k^{(t-1)}) = \left(D_u U_f(X^{(t-1)})\right)^T \text{vec}(Y - (X^{(t-1)} \otimes K^{(t-1)}) d_s),$$

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

Implementation of $D_u U_f(\cdot)$: With Pytorch 4 framework, we can directly perform “torch.nn.function.unfold” with stride $= s$ on $X^{(t-1)} \in \mathbb{R}^{HW \times W}$ to get $(D_u U_f(X^{(t-1)}))^T \in \mathbb{R}^{p^2 \times hw}$, and execute “torch.permute” to get $D_u U_f(X^{(t-1)}) \in \mathbb{R}^{hw \times p^2}$.

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

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

(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 $\text{proxNet}_{\theta}(\cdot)$. (2) The exploited ResNet for the proximal network $\text{proxNet}_{\theta^2}(\cdot)$.

where $h(X^{(t-1)}) = \|Y - (X^{(t-1)} \otimes K^{(t)}) \downarrow_s\|^2_F$; $\nabla h (X^{(t-1)}) = K^{(t)} \otimes_T (Y - (X^{(t-1)} \otimes K^{(t)}) \downarrow_s)$; $\delta_2$ denotes the stepsize parameter. Then the equivalent optimization problem is:

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

(8)

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

$$X^{(t)} = \text{prox}_{\lambda_2 \delta_2} \left( X^{(t-1)} - \delta_2 K^{(t)} \otimes_T \left( Y - (X^{(t-1)} \otimes K^{(t)}) \downarrow_s \right) \right),$$

(9)

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

2 Details of Network Module Design

In this section, we provide more details of network design, including the gradient adjuster, $\text{proxNet}_{\theta}(\cdot)$ and $\text{proxNet}_{\theta^2}(\cdot)$.

Gradient adjuster. As stated in Section 4.1 of the main text, we adopt an adjuster to the gradient $G^{(t)}$, 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] PSNR | Urban100 [7] SSIM | BSD100 [10] PSNR | BSD100 [10] SSIM | Set14 [14] PSNR | Set14 [14] SSIM | Set5 [3] PSNR | Set5 [3] SSIM |
|-----------|-------|------------------|------------------|------------------|------------------|----------------|----------------|----------------|----------------|
| KXNet*    | 0     | 27.55            | 0.8425           | 29.71            | 0.8293           | 30.45          | 0.8532         | 33.63          | 0.9213         |
| KXNet(ours) | 0    | **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) | 5    | 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) | 15   | 25.45            | 0.7500           | 26.87            | 0.6959           | 27.59          | 0.7422         | 29.93          | **0.8449**     |

Fig. 1, the residual image \( E_t^{(t)} \) are deconvolved with blur kernel \( K^{(t)} \) to obtain the gradient \( G_x^{(t)} \). We can clearly find that due to the “uneven overlap” phenomenon with transposed convolution, the obtained gradient \( G_x^{(t)} \) 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 \( K^{(t)} \otimes_s T^{(t)} \mathbf{1} \) to calculate the degree of uneven overlap and element-wisely divide \( G_x^{(t)} \) with \( K^{(t)} \otimes_s T^{(t)} \mathbf{1} \) to get an adjusted gradient, \( \hat{G}_x^{(t)} \). As illustrated in Fig. 1, \( \hat{G}_x^{(t)} \) has more precise textures and edges than \( G_x^{(t)} \), which will improve the recovery performance of X-net.

**Proximal Network Architecture.** As stated in the main text, the proximal network \( \text{proxNet}^{(t)} \) and \( \text{proxNet}^{(t)}(\cdot) \) \( (t = 0, 1, \cdots, T) \) are two shallow ResNets. Fig. 2 shows the architectural details of \( \text{proxNet}^{(t)} \) and \( \text{proxNet}^{(t)}(\cdot) \), respectively. For each Resblock in each stage in \( \text{proxNet}^{(t)}(\cdot) \), we simply adopt the same structure. \( \text{proxNet}^{(t)}(\cdot) \) 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 \( X^{(t-1)} \) and \( \hat{G}_x^{(t)} \) as the input of the proximal network \( \text{proxNet}^{(t)}(\cdot) \). 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 \( X^{(t-1)} \) and \( \hat{G}_x^{(t)} \) as the input of the proximal network \( \text{proxNet}^{(t)}(\cdot) \), 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 Level |
|--------------|-------|--------------|-------------|------------|---------|-------------|
|              |       | 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]    | x4    | 23.22 | 0.6791 | 26.03 | 0.6896 | 25.92 | 0.7217 | 27.85 | 0.8095 |
| IKC [6]      | x3    | 27.46 | 0.6801 | 29.85 | 0.8390 | 30.69 | 0.8614 | 33.99 | 0.9220 |
| 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 |

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.
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 sub-
stantiates 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)