Plug-and-Play Image Restoration With Deep Denoiser Prior

Kai Zhang, Yawei Li, Wangmeng Zuo, *Senior Member, IEEE*, Lei Zhang, *Fellow, IEEE*, Luc Van Gool, and Radu Timofte, *Member, IEEE*

Abstract—Recent works on plug-and-play image restoration have shown that a denoiser can implicitly serve as the image prior for model-based methods to solve many inverse problems. Such a property induces considerable advantages for plug-and-play image restoration (e.g., integrating the flexibility of model-based method and effectiveness of learning-based methods) when the denoiser is discriminatively learned via deep convolutional neural network (CNN) with large modeling capacity. However, while deeper and larger CNN models are rapidly gaining popularity, existing plug-and-play image restoration hinders its performance due to the lack of suitable denoiser prior. In order to push the limits of plug-and-play image restoration, we set up a benchmark deep denoiser prior by training a highly flexible and effective CNN denoiser. We then plug the deep denoiser prior as a modular part into a half quadratic splitting based iterative algorithm to solve various image restoration problems. We, meanwhile, provide a thorough analysis of parameter setting, intermediate results and empirical convergence to better understand the working mechanism. Experimental results on three representative image restoration tasks, including deblurring, super-resolution and demosaicing, demonstrate that the proposed plug-and-play image restoration with deep denoiser prior not only significantly outperforms other state-of-the-art model-based methods but also achieves competitive or even superior performance against state-of-the-art learning-based methods. The source code is available at https://github.com/cszn/DPIR.

Index Terms—Denoiser prior, image restoration, convolutional neural network, half quadratic splitting, plug-and-play

1 Introduction

Image restoration (IR) has been a long-standing problem for its highly practical value in various low-level vision applications [1], [2]. In general, the purpose of image restoration is to recover the latent clean image $\mathbf{x}$ from its degraded observation $\mathbf{y} = \mathcal{T}(\mathbf{x}) + \mathbf{n}$, where $\mathcal{T}$ is the noise-irrelevant degradation operation, $\mathbf{n}$ is assumed to be additive white Gaussian noise (AWGN) of standard deviation σ. By specifying different degradation operations, one can correspondingly get different IR tasks. Typical IR tasks would be image denoising when $\mathcal{T}$ is an identity operation, image deblurring when $\mathcal{T}$ is a two-dimensional convolution operation, image super-resolution when $\mathcal{T}$ is a composite operation of convolution and down-sampling, color image demosaicing when $\mathcal{T}$ is a color filter array (CFA) masking operation.

- *Kai Zhang, Yawei Li, and Radu Timofte are with the Computer Vision Lab, ETH Zürich, 8092 Zürich, Switzerland. E-mail: {kai.zhang, yawei.li, radu.timofte}@vision.ee.ethz.ch.*
- *Wangmeng Zuo is with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China. E-mail: cswmzuo@gmail.com.*
- *Lei Zhang is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. E-mail: cslzhang@comp.polyu.edu.hk.*
- *Luc Van Gool is with the Computer Vision Lab, ETH Zürich, 8092 Zürich, Switzerland, and also with the KU Leuven, 3000 Leuven, Belgium. E-mail: vangool@vision.ee.ethz.ch.*

Manuscript received 18 Sept. 2020; revised 17 May 2021; accepted 3 June 2021. Date of publication 14 June 2021; date of current version 9 Sept. 2022. (Corresponding author: Radu Timofte.) Recommended for acceptance by C. C. Loy. Digital Object Identifier no. 10.1109/TPAMI.2021.3088914

Since IR is an ill-posed inverse problem, the prior which is also called regularization needs to be adopted to constrain the solution space [3], [4]. From a Bayesian perspective, the solution $\hat{\mathbf{x}}$ can be obtained by solving a Maximum A Posteriori (MAP) estimation problem

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}) + \log p(\mathbf{x}), \tag{1}$$

where $\log p(\mathbf{y}|\mathbf{x})$ represents the log-likelihood of observation $\mathbf{y}$, $\log p(\mathbf{x})$ delivers the prior of clean image $\mathbf{x}$ and is independent of degraded image $\mathbf{y}$. More formally, (1) can be reformulated as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2} \|\mathbf{y} - \mathcal{T}(\mathbf{x})\|^2 + \lambda \mathcal{R}(\mathbf{x}), \tag{2}$$

where the solution minimizes an energy function composed of a data term $\frac{1}{2\sigma^2}\|\mathbf{y} - \mathcal{T}(\mathbf{x})\|^2$ and a regularization term $\lambda\mathcal{R}(\mathbf{x})$ with regularization parameter λ. Specifically, the data term guarantees the solution accords with the degradation process, while the prior term alleviates the ill-posedness by enforcing desired property on the solution.

Generally, the methods to solve (2) can be divided into two main categories, i.e., model-based methods and learning-based methods. The former aim to directly solve (2) with some optimization algorithms, while the latter mostly train a truncated unfolding inference through an optimization of a loss function on a training set containing N degraded-clean image pairs $\{(\mathbf{y}_i, \mathbf{x}_i)\}_{i=1}^{N}$ [5], [6], [7], [8], [9]. In particular, the learning-based methods are usually modeled as the following bi-level optimization problem

$$\begin{cases} \min\limits_{\Theta} \sum\limits_{i=1}^{N} \mathcal{L}(\hat{\mathbf{x}}_i, \mathbf{x}_i) \\ s.t. \quad \hat{\mathbf{x}}_i = \arg\min\limits_{\mathbf{x}} \frac{1}{2\sigma^2} \|\mathbf{y}_i - \mathcal{T}(\mathbf{x})\|^2 + \lambda \mathcal{R}(\mathbf{x}), \end{cases} \tag{3}$$

where Θ denotes the trainable parameters, $\mathcal{L}(\hat{\mathbf{x}}_i, \mathbf{x}_i)$ measures the loss of estimated clean image $\hat{\mathbf{x}}_i$ with respect to ground truth image $\mathbf{x}_i$. By replacing the unfolding inference (3b) with a predefined function $\hat{\mathbf{x}} = f(\mathbf{y}, \Theta)$, one can treat the plain learning-based methods as general case of (3).

It is easy to note that one main difference between model-based methods and learning-based methods is that, the former are flexible to handle various IR tasks by simply specifying $\mathcal{T}$ and can directly optimize on the degraded image $\mathbf{y}$, whereas the later require cumbersome training to learn the model before testing and are usually restricted by specialized tasks. Nevertheless, learning-based methods can not only enjoy a fast testing speed but also tend to deliver better performance due to the end-to-end training. In contrast, model-based methods are usually time-consuming with sophisticated priors for the purpose of good performance [10]. As a result, these two categories of methods have their respective merits and drawbacks, and thus it would be attractive to investigate their integration which leverages their respective merits. Such an integration has resulted in the deep plug-and-play IR method which replaces the denoising subproblem of model-based optimization with learning-based CNN denoiser prior.

The main idea of deep plug-and-play IR is that, with the aid of variable splitting algorithms, such as alternating direction method of multipliers (ADMM) [11] and half-quadratic splitting (HQS) [12], it is possible to deal with the data term and prior term separately [13], and particularly, the prior term only corresponds to a denoising subproblem [14], [15], [16] which can be solved via deep CNN denoiser. Although several deep plug-and-play IR works have been proposed, they typically suffer from the following drawbacks. First, they either adopt different denoisers to cover a wide range of noise levels or use a single denoiser trained on a certain noise level, which are not suitable to solve the denoising subproblem. For example, the IRCNN [17] denoisers involve 25 separate 7-layer denoisers, each of which is trained on an interval noise level of 2. Second, their deep denoisers are not powerful enough, and thus, the performance limit of deep plug-and-play IR is unclear. Third, a deep empirical understanding of their working mechanism is lacking.

This paper is an extension of our previous work [17] with a more flexible and powerful deep CNN denoiser which aims to push the limits of deep plug-and-play IR by conducting extensive experiments on different IR tasks. Specifically, inspired by FFDNet [18], the proposed deep denoiser can handle a wide range of noise levels via a single model by taking the noise level map as input. Moreover, its effectiveness is enhanced by taking advantages of both ResNet [19] and U-Net [20]. The deep denoiser is further incorporated into HQS-based plug-and-play IR to show the merits of using powerful deep denoiser. Meanwhile, a novel periodical geometric self-ensemble is proposed to potentially improve the performance without introducing extra computational burden, and a thorough analysis of parameter setting, intermediate results and empirical convergence are provided to better understand the working mechanism of the proposed deep plug-and-play IR.

The contribution of this work is summarized as follows:

- A flexible and powerful deep CNN denoiser is trained. It not only outperforms the state-of-the-art deep Gaussian denoising models but also is suitable to solve plug-and-play IR.
- The HQS-based plug-and-play IR is thoroughly analyzed with respect to parameter setting, intermediate results and empirical convergence, providing a better understanding of the working mechanism.
- Extensive experimental results on deblurring, super-resolution and demosaicing have demonstrated the superiority of the proposed plug-and-play IR with deep denoiser prior.

2 Related Works

Plug-and-play IR generally involves two steps. The first step is to decouple the data term and prior term of the objective function via a certain variable splitting algorithm, resulting in an iterative scheme consisting of alternately solving a data subproblem and a prior subproblem. The second step is to solve the prior subproblem with any off-the-shelf denoisers, such as K-SVD [21], non-local means [22], BM3D [23]. As a result, unlike traditional model-based methods which needs to specify the explicit and hand-crafted image priors, plug-and-play IR can implicitly define the prior via the denoiser. Such an advantage offers the possibility of leveraging very deep CNN denoiser to improve effectiveness.

2.1 Plug-and-Play IR With Non-CNN Denoiser

The plug-and-play IR can be traced back to [4], [14], [16]. In [24], Danielyan *et al.* used Nash equilibrium to derive an iterative decoupled deblurring BM3D (IDDBM3D) method for image debluring. In [25], a similar method equipped with CBM3D denoiser prior was proposed for single image super-resolution (SISR). By iteratively updating a back-projection step and a CBM3D denoising step, the method has an encouraging performance for its PSNR improvement over SRCNN [26]. In [14], the augmented Lagrangian method was adopted to fuse the BM3D denoiser to solve image deblurring task. With a similar iterative scheme as in [24], the first work that treats the denoiser as "plug-and-play prior" was proposed in [16]. Prior to that, a similar plug-and-play idea is mentioned in [4] where HQS algorithm is adopted for image denoising, deblurring and inpainting. In [15], Heide *et al.* used an alternative to ADMM and HQS, i.e., the primal-dual algorithm [27], to decouple the data term and prior term. In [28], Teodoro *et al.* plugged class-specific Gaussian mixture model (GMM) denoiser [4] into ADMM to solve image deblurring and compressive imaging. In [29], Metzler *et al.* developed a denoising-based approximate message passing (AMP) method to integrate denoisers, such as BLS-GSM [30] and BM3D, for compressed sensing reconstruction. In [31], Chan *et al.* proposed plug-and-play ADMM algorithm with BM3D denoiser for single image super-resolution and quantized Poisson image recovery for single-photon imaging. In [32],

Kamilov *et al.* proposed fast iterative shrinkage thresholding algorithm (FISTA) with BM3D and WNNM [10] denoisers for nonlinear inverse scattering. In [33], Sun *et al.* proposed FISTA by plugging TV and BM3D denoiser prior for Fourier ptychographic microscopy. In [34], Yair and Michaeli proposed to use WNNM denoiser as the plug-and-play prior for inpainting and deblurring. In [35], Gavaskar and Chaudhury investigated the convergence of ISTA-based plug-and-play IR with non-local means denoiser.

2.2 Plug-and-Play IR With Deep CNN Denoiser

With the development of deep learning techniques such as network design and gradient-based optimization algorithm, CNN-based denoiser has shown promising performance in terms of effectiveness and efficiency. Following its success, a flurry of CNN denoiser based plug-and-play IR works have been proposed. In [37], Romano *et al.* proposed explicit regularization by TNRD denoiser for image deblurring and SISR. In our previous work [17], different CNN denoisers are trained to plug into HQS algorithm to solve deblurring and SISR. In [38], Tirer and Giryes proposed iterative denoising and backward projections with IRCNN denoisers for image inpainting and deblurring. In [39], Gu *et al.* proposed to adopt WNNM and IRCNN denoisers for plug-and-play deblurring and SISR. In [40], Tirer and Giryes proposed use the IRCNN denoisers for plug-and-play SISR. In [41], Li and Wu plugged the IRCNN denoisers into the split Bregman iteration algorithm to solve depth image inpainting. In [42], Ryu *et al.* provided the theoretical convergence analysis of plug-and-play IR based on forward-backward splitting algorithm and ADMM algorithm, and proposed spectral normalization to train a DnCNN denoiser. In [43], Sun *et al.* proposed a block coordinate regularization-by-denoising (RED) algorithm by leveraging DnCNN [44] denoiser as the explicit regularizer.

Although plug-and-play IR can leverage the powerful expressiveness of CNN denoiser, existing methods generally exploit DnCNN or IRCNN which do not make full use of CNN. Typically, the denoiser for plug-and-play IR should be non-blind and requires to handle a wide range of noise levels. However, DnCNN needs to separately learn a model for each noise level. To reduce the number of denoisers, some works adopt one denoiser fixed to a small noise level. However, according to [37] and as will be shown in Section 5.1.3, such a strategy tends to require a large number of iterations for a satisfying performance which would increase the computational burden. While IRCNN denoisers can handle a wide range of noise levels, it consists of 25 separate 7-layer denoisers, among which each denoiser is trained on an interval noise level of 2. Such a denoiser suffers from two drawbacks. First, it does not have the flexibility to hand a specific noise level. Second, it is not effective enough due to the shallow layers. Given the above considerations, it is necessary to devise a flexible and powerful denoiser to boost the performance of plug-and-play IR.

2.3 Why not Use a Blind Gaussian Denoiser for Plug-and-Play IR?

It is worth to emphasize that the denoiser for plug-and-play IR should be designed for non-blind Gaussian denoising. The reason is two-fold. First, as will be shown in (6b) of the iterative solution for plug-and-play IR, the sub-problem actually corresponds to a non-blind Gaussian denoising problem with a Gaussian noise level. Second, although the non-blind Gaussian denoising problem could be solved via the blind Gaussian denoiser, the role of the Gaussian denoiser for plug-and-play IR is to smooth out the unknown noise (e.g., structural noise introduced during iterations) rather than remove the Gaussian noise. As will be shown in Section 6, the noise distribution during iterations is usually non-Gaussian and varies across different IR tasks and even different iterations. Moreover, as we will see in Section 5.1.4, while the non-blind Gaussian denoiser can smooth out such non-Gaussian noise by setting a proper noise level, the blind Gaussian denoiser does not have such ability as it can only remove the Gaussian-like noise [18].

2.4 Difference to Deep Unfolding IR

It should be noted that, apart from plug-and-play IR, deep unfolding IR [45], [46], [47], [48] can also incorporate the advantages of both model-based methods and learning-based methods. The main difference between them is that the latter interprets a truncated unfolding optimization as an end-to-end trainable deep network and thus usually produce better results with fewer iterations. However, deep unfolding IR needs separate training for each task. On the contrary, plug-and-play IR is easy to deploy without such additional training.

3 Learning Deep CNN Denoiser Prior

Although various CNN-based denoising methods have been recently proposed, most of them are not designed for plug-and-play IR. In [50], [51], [52], a novel training strategy without ground-truth is proposed. In [53], [54], [55], [56], real noise synthesis technique is proposed to handle real digital photographs. However, from a Bayesian perspective, the denoiser for plug-and-play IR should be a Gaussian denoiser. Hence, one can add synthetic Gaussian noise to clean image for supervised training. In [57], [58], [59], [60], the non-local module was incorporated into the network design for better restoration. However, these methods learn a separate model for each noise level. Perhaps the most suitable denoiser for plug-and-play IR is FFDNet [18] which can handle a wide range of noise levels by taking the noise level map as input. Nevertheless, FFDNet only has a comparable performance to DnCNN and IRCNN, thus lacking effectiveness to boost the performance of plug-and-play IR. For this reason, we propose to improve FFDNet by taking advantage of the widely-used U-Net [20] and ResNet [19] for architecture design.

3.1 Denoising Network Architecture

It is well-known that U-Net [20] is effective and efficient for image-to-image translation, while ResNet [19] is superior in increasing the modeling capacity by stacking multiple residual blocks. Following FFDNet [18] that takes the noise level map as input, the proposed denoiser, namely DRUNet, further integrates residual blocks into U-Net for effective denoiser prior modeling. Note that this work focuses on providing a flexible and powerful pre-trained denoiser to

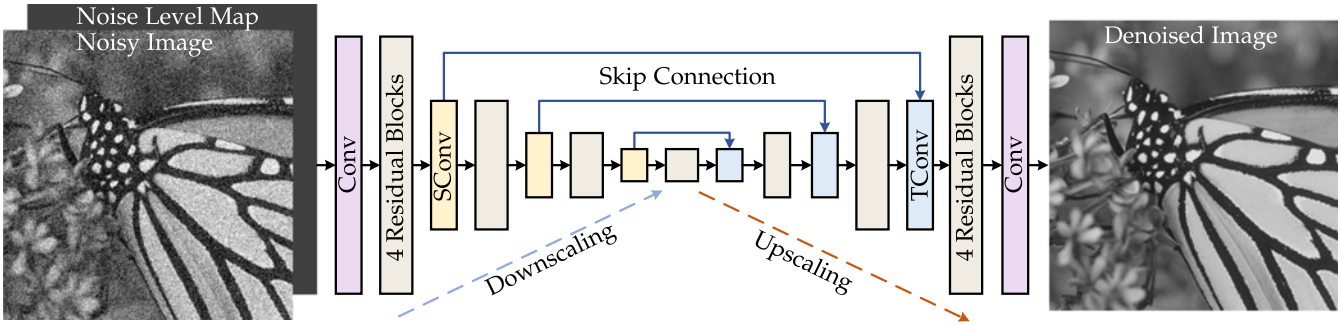


Fig. 1. The architecture of the proposed DRUNet denoiser prior. DRUNet takes an additional noise level map as input and combines U-Net [20] and ResNet [36]. "SConv" and "TConv" represent strided convolution and transposed convolution, respectively.

benefit existing plug-and-play IR methods rather than designing new denoising network architecture. Actually, the similar idea of combining U-Net and ResNet can also be found in other works such as [61], [62].

The architecture of DRUNet is illustrated in Fig. 1. Like FFDNet, DRUNet has the ability to handle various noise levels via a single model. The backbone of DRUNet is U-Net which consists of four scales. Each scale has an identity skip connection between 2×2 strided convolution (SConv) downscaling and 2×2 transposed convolution (TConv) upscaling operations. The number of channels in each layer from the first scale to the fourth scale are 64, 128, 256 and 512, respectively. Four successive residual blocks are adopted in the downscaling and upscaling of each scale. Inspired by the network architecture design for super-resolution in [63], no activation function is followed by the first and the last convolutional (Conv) layers, as well as SConv and TConv layers. In addition, each residual block only contains one ReLU activation function.

It is worth noting that the proposed DRUNet is bias-free, which means no bias is used in all the Conv, SConv and TConv layers. The reason is two-fold. First, bias-free network with ReLU activation and identity skip connection naturally enforces scaling invariance property of many image restoration tasks, i.e., $f(ax) = af(x)$ holds true for any scalar $a \geq 0$ (please refer to [64] for more details). Second, we have empirically observed that, for the network with bias, the magnitude of bias would be much larger than that of filters, which in turn may harm the generalizability.

3.2 Training Details

It is well known that CNN benefits from the availability of large-scale training data. To enrich the denoiser prior for plug-and-play IR, instead of training on a small dataset that includes 400 Berkeley segmentation dataset (BSD) images of size 180×180 [9], we construct a large dataset consisting of 400 BSD images, 4,744 images of Waterloo Exploration Database [65], 900 images from DIV2K dataset [66], and 2,750 images from Flick2K dataset [63]. Because such a dataset covers a larger image space, the learned model can slightly improve the PSNR results on BSD68 dataset [3] while having an obvious PSNR gain on testing datasets from a different domain.

As a common setting for Gaussian denoising, the noisy counterpart $\mathbf{y}$ of clean image $\mathbf{x}$ is obtained by adding AWGN with noise level σ. Correspondingly, the noise level map is a uniform map filled with σ and has the same spatial size as noisy image. To handle a wide range of noise levels, the noise level σ is randomly chosen from $[0, 50]$ during training. Note that the noisy images are not clipped into the range of $[0, 255]$. The reason is that the clipping operation would change the distribution of the noise, which in turn will give rise to inaccurate solution for plug-and-play IR. The network parameters are optimized by minimizing the L1 loss rather than L2 loss between the denoised image and its ground-truth with Adam algorithm [67]. Although there is no direct evidence on which loss would result in better performance, it is widely acknowledged that L1 loss is more robust than L2 loss in handling outliers [68]. Regarding to denoising, outliers may occur during the sampling of AWGN. In this sense, L1 loss tends to be more stable than L2 loss for denoising network training. The learning rate starts from 1e-4 and then decreases by half every 100,000 iterations and finally ends once it is smaller than 5e-7. In each iteration during training, 16 patches with patch size of 128×128 were randomly sampled from the training data. We separately learn a denoiser model for grayscale image and color image. It takes about four days to train the model with PyTorch and an Nvidia Titan Xp GPU.

3.3 Denoising Results

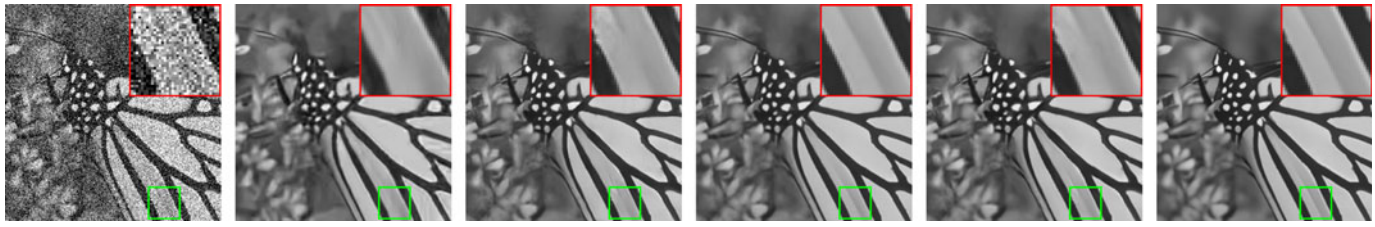
3.3.1 *Grayscale Image Denoising*

For grayscale image denoising, we compared the proposed DRUNet denoiser with several state-of-the-art denoising methods, including two representative model-based methods (i.e., BM3D [23] and WNNM [10]), five CNN-based methods which separately learn a single model for each noise level (i.e., DnCNN [44], N^3Net [60], NLRN [59], RNAN [69], FOCNet [70]) and two CNN-based methods which were trained to handle a wide range of noise levels (i.e., IRCNN [17] and FFDNet [18]). Note that N^3Net, NLRN and RNAN adopt non-local module in the network architecture design so as to exploit non-local image prior. The PSNR results of different methods on the widely-used Set12 [44] and BSD68 [3], [49] datasets for noise levels 15, 25 and 50 are reported in Table 1. It can be seen that DRUNet achieves the best PSNR results for all the noise levels on the two datasets. Specifically, DRUNet has an average PSNR gain of about 0.9dB over BM3D and surpasses DnCNN, IRCNN and FFDNet by an average PSNR of 0.5dB on Set12 dataset and 0.25dB on BSD68 dataset. Despite the fact that NLRN, RNAN and FOCNet learn a separate model for each noise level and have a very competitive performance, they fail to

TABLE 1
Average PSNR(dB) Results of Different Methods With Noise Levels 15, 25 and 50 on the Widely-Used Set12 and BSD68 [3], [44], [49] Datasets

Datasets	Noise Level	BM3D	WNNM	DnCNN	N^3Net	NLRN	RNAN	FOCNet	IRCNN	FFDNet	DRUNet
Set12	15	32.37	32.70	32.86	–	33.16	–	33.07	32.77	32.75	33.25
	25	29.97	30.28	30.44	30.55	30.80	–	30.73	30.38	30.43	30.94
	50	26.72	27.05	27.18	27.43	27.64	27.70	27.68	27.14	27.32	27.90
BSD68	15	31.08	31.37	31.73	–	31.88	–	31.83	31.63	31.63	31.91
	25	28.57	28.83	29.23	29.30	29.41	–	29.38	29.15	29.19	29.48
	50	25.60	25.87	26.23	26.39	26.47	26.48	26.50	26.19	26.29	26.59

The best and second best results are highlighted in red and blue colors, respectively.



(a) Noisy (14.78dB) (b) BM3D (25.82dB) (c) DnCNN (26.83dB) (d) RNAN (27.18dB) (e) FFDNet (26.92dB) (f) DRUNet (27.31dB)

Fig. 2. Grayscale image denoising results of different methods on image "*Monarch*" from Set12 dataset with noise level 50.

outperform DRUNet. Fig. 2 shows the grayscale image denoising results of different methods on image "*Monarch*" from Set12 dataset with noise level 50. It can be seen that DRUNet can recover much sharper edges than BM3D, DnCNN, FFDNet while having similar result with RNAN.

3.3.2 Color Image Denoising

Since existing methods mainly focus on grayscale image denoising, we only compare DRUNet with CBM3D, DnCNN, IRCNN and FFDNet for color denoising. Table 2 reports the color image denoising results of different methods for noise levels 15, 25 and 50 on CBSD68 [3], [44], [49], Kodak24 [71] and McMaster [72] datasets. One can see that DRUNet outperforms the other competing methods by a large margin. It is worth noting that while having a good performance on CBSD68 dataset, DnCNN does not perform well on McMaster dataset. Such a discrepancy highlights the importance of reducing the image domain gap between training and testing for image denoising. The visual results of different methods on image "*163085*" from CBSD68 dataset with noise level 50 are shown in Fig. 3 from which it can be seen that DRUNet can recover more fine details and textures than the competing methods.

TABLE 2
Average PSNR(dB) Results of Different Methods for Noise Levels 15, 25 and 50 on CBSD68 [3], [44], [49], Kodak24, and McMaster Datasets

Datasets	Noise Level	CBM3D	DnCNN	IRCNN	FFDNet	DRUNet
CBSD68	15	33.52	33.90	33.86	33.87	34.30
	25	30.71	31.24	31.16	31.21	31.69
	50	27.38	27.95	27.86	27.96	28.51
Kodak24	15	34.28	34.60	34.69	34.63	35.31
	25	32.15	32.14	32.18	32.13	32.89
	50	28.46	28.95	28.93	28.98	29.86
McMaster	15	34.06	33.45	34.58	34.66	35.40
	25	31.66	31.52	32.18	32.35	33.14
	50	28.51	28.62	28.91	29.18	30.08

The best and second best results are highlighted in red and blue colors, respectively.

3.3.3 Extended Application to JPEG Image Deblocking

The proposed DRUNet is also applicable to remove other different noise types, such as JPEG compression artifacts. By simply changing the training data and replacing the noise level σ of AWGN with quality factor q of JPEG compression, a DRUNet model for JPEG image deblocking is trained. We set the quality factor range to $[10, 95]$, where quality factor 10 represents lower quality and higher compression, and vice versa. Specifically, the quality factor q are normalized with $(100 - q)/100$ for 0-1 normalization. We use the same training data as in denoising for training. Table 3 reports the average PSNR (dB) results of different methods for JPEG image deblocking with quality factors 10, 20, 30 and 40 on Classic5 and LIVE1 datasets. The compared methods include ARCNN [73], TNRD [9], DnCNN3 [44] and RNAN [58], QGAC [74]. Since DnCNN3 is trained also for denosing and SISR, we re-trained a non-blind DnCNN3 model with our training data. Compared to the original DnCNN3 model, the new model has average PSNR gains of 0.21dB and 0.19dB on the two testing datasets. To quantify the performance contribution of training data, we also trained a DRUNet model with less training data as in [18]. The results show that the PSNR decreases by 0.04dB on average, which demonstrates that a large training data can slightly improve the performance for JPEG image deblocking. From Table 3, we can see that DRUNet outperforms ARCNN, TNRD, DnCNN3 and QGAC by a large margin and has an average PSNR gain

(a) Noisy (14.99dB) (b) BM3D (28.36dB) (c) DnCNN (28.68dB) (d) FFDNet (28.75dB) (e) IRCNN (28.69dB) (f) DRUNet (29.28dB)

Fig. 3. Color image denoising results of different methods on image "*163085*" from CBSD68 dataset with noise level 50.

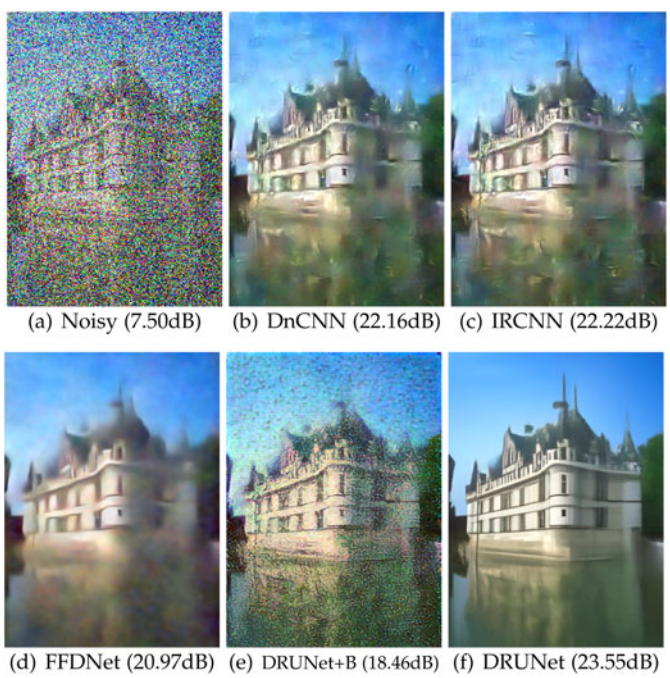
of 0.15dB over RNAN, which further demonstrates the flexibility and effectiveness of the proposed DRUNet.

3.3.4 Generalizability to Unseen Noise Level

In order to show the advantage of the bias-free DRUNet, we also train a DRUNet+B model whose biases were randomly initialized from a uniform distribution in $[-1, 1]$. Fig. 4 provides the visual results comparison between different models on a noisy image with an extremely large unseen noise level of 200. Note that since DnCNN and IRCNN do not have the flexibility to change the noise level, we first multiply the noisy image by a factor of 0.25 so that the noise level changes from 200 to 50. We then apply the DnCNN and IRCNN models for denoising and finally obtain the denoising results with a multiplication of 4. From Fig. 4, we can seen that, even trained on noise level range of $[0, 50]$, the bias-free DRUNet can still perform well, whereas DRUNet+B (with biases) introduces noticeable visual artifacts while having a much lower PSNR. As we will see in Section 3.3.5, DRUNet+B has a comparable performance with bias-free DRUNet for noise levels in $[0, 50]$. Thus, we can conclude that bias-free DRUNet can enhance the generalizability to unseen noise level. Note that whether bias-free network can benefit other tasks or not remains further study.

TABLE 3
Average PSNR(dB) Results of Different Methods for JPEG Image Deblocking With Quality Factors 10, 20, 30 and 40 on Classic5 and LIVE1 Datasets

Datasets	q	ARCNN	TNRD	DnCNN3	RNAN	QGAC	DRUNet
Classic5	10	29.03	29.28	29.40	29.96	29.84	30.16
	20	31.15	31.47	31.63	32.11	31.98	32.39
	30	32.51	32.78	32.91	33.38	33.22	33.59
	40	33.34	-	33.77	34.27	34.05	34.41
LIVE1	10	28.96	29.15	29.19	29.63	29.51	29.79
	20	31.29	31.46	31.59	32.03	31.83	32.17
	30	32.67	32.84	32.98	33.45	33.20	33.59
	40	33.63	-	33.96	34.47	34.16	34.58



(a) Noisy (7.50dB) (b) DnCNN (22.16dB) (c) IRCNN (22.22dB)

(d) FFDNet (20.97dB) (e) DRUNet+B (18.46dB) (f) DRUNet (23.55dB)

Fig. 4. An example to show the generalizability advantage of the proposed bias-free DRUNet. The noise level of the noisy image is 200.

3.3.5 Ablation Study

In order to further analyze the proposed DRUNet, we have performed an ablation study to quantify the performance contribution of different factors such as residual blocks, training data, biases, and noise level map. Table 4 reports the comparisons between DRUNet with four different cases, including case 1: DRUNet without skip connections of the residual blocks, case 2: DRUNet with less training data as in [18], case 3: DRUNet without removing the biases (i.e., DRUNet+B), and case 4: DRUNet without taking noise level map as input. By comparison, we can have the following conclusions. First, the residual blocks can ease the training to get a better performance. Second, the performance on Set12 dataset tends to be saturated with enough training data as the DRUNet model with more training data only improves the PSNR by an average of 0.01dB. Third, DRUNet with biases has a similar performance to bias-free DRUNet for the trained noise levels, however, the bias-free DRUNet can enhance the generalizability to unseen noise level. Fourth, the noise level map can improve the

TABLE 4
The PSNR Results Comparison With Different Cases (i.e, Case 1: DRUNet Without Residual Blocks, Case 2: DRUNet With Less Training Data as in [18], Case 3: DRUNet Without Removing the Biases, and Case 4: DRUNet Without Taking Noise Level Map as Input) With Noise Levels 15, 25 and 50 on Set12 Dataset

Datasets	Noise Level	Case 1	Case 2	Case 3	Case 4	DRUNet
Set12	15	33.12	33.23	33.24	33.16	33.25
	25	30.82	30.92	30.93	30.86	30.94
	50	27.78	27.87	27.89	27.83	27.90

TABLE 5
Runtime (in Seconds), FLOPs (in G) and Max GPU Memory (in GB) of Different Methods on Images of Size 256×256 and 512×512 With Noise Level 50

Metric	Image Size	DnCNN	IRCNN	FFDNet	RNAN	RNAN*	DRUNet
Runtime	256×256	0.0087	0.0066	0.0023	0.4675	0.4662	0.0221
	512×512	0.0314	0.0257	0.0071	2.1530	1.8769	0.0733
FLOPs	256×256	36.38	12.18	7.95	774.67	495.79	102.91
	512×512	145.52	48.72	31.80	3416.29	2509.92	411.65
Memory	256×256	0.0339	0.0346	0.0114	0.3525	2.9373	0.2143
	512×512	0.1284	0.1360	0.0385	0.4240	3.2826	0.4911

The experiments are conducted in PyTorch on a PC with an Intel Xeon 3.5GHz 4-core CPU, 4-8GB of RAM and an Nvidia Titan Xp GPU.

performance as it introduces extra information of the noise. Such a phenomenon has also been reported in [18].

3.3.6 Runtime, FLOPs and Maximum GPU Memory Consumption

Table 5 reports the runtime, FLOPs and maximum GPU memory consumption comparison with four representative methods (i.e., DnCNN, IRCNN, FFDNet and RNAN) on images of size 256×256 and 512×512 with noise level 50. We do not report other methods such as FOCNet for comparison since they are not implemented in PyTorch for a fair and easy comparison. Note that, in order to reduce the memory caused by the non-local module, RNAN splits the input image into overlapped patches with predefined maximum spatial size and then aggravates the results to obtain the final denoised image. The default maximum spatial size is 10,000 which is equivalent to a size of 100×100. We also compare RNAN* which sets maximum spatial size to 70,000. As a simple example, RNAN and RNAN* splits an image of size 512×512 into 64 and 4 overlapped patches, respectively. It should be noted that NLRN which also adopts a similar non-local module as RNAN uses a different strategy reduce the memory, i.e, fixing the patch size to 43×43. However, it uses a small stride of 7 which would largely increase the computational burden.

From Table 5, one can see that FFDNet achieves the best performance on runtime, FLOPs and memory. Compared to DnCNN, DRUNet has much better PSNR values and only doubles the runtime, triples the FLOPs, and quadruples the maximum GPU memory consumption. In contrast, RNAN is about 60 times slower than DnCNN and also has a much larger FLOPs. In addition, it would dramatically aggravate the maximum GPU memory consumption with the increase of the predefined maximum spatial size. Note that RNAN does not outperform DRUNet in terms of PSNR. Such a phenomenon highlights that the non-local module in RNAN may not be a proper way to improve PSNR and further study is required to improve the runtime, FLOPs and maximum GPU memory consumption.

According to the above results, we can conclude that DRUNet is a flexible and powerful denoiser prior for plug-and-play IR.

4 HQS Algorithm for Plug-and-Play IR

Although there exist various variable splitting algorithms for plug-and-play IR, HQS owes its popularity to the simplicity and fast convergence. We therefore adopt HQS in this paper. On the other hand, there is no doubt that parameter setting is always a non-trivial issue [37]. In other words, careful parameter setting is needed to obtain a good performance. To have a better understanding on the HQS-based plug-and-play IR, we will discuss the general methodology for parameter setting after providing the HQS algorithm. We then propose a periodical geometric self-ensemble strategy to potentially improve the performance.

4.1 Half Quadratic Splitting (HQS) Algorithm

In order to decouple the data term and prior term of (2), HQS first introduces an auxiliary variable $\mathbf{z}$, resulting in a constrained optimization problem given by

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2}\|\mathbf{y} - \mathcal{T}(\mathbf{x})\|^2 + \lambda\mathcal{R}(\mathbf{z}) \quad s.t. \quad \mathbf{z} = \mathbf{x}. \tag{4}$$

(4) is then solved by minimizing the following problem

$$\mathcal{L}_\mu(\mathbf{x}, \mathbf{z}) = \frac{1}{2\sigma^2}\|\mathbf{y} - \mathcal{T}(\mathbf{x})\|^2 + \lambda\mathcal{R}(\mathbf{z}) + \frac{\mu}{2}\|\mathbf{z} - \mathbf{x}\|^2, \tag{5}$$

where μ is a penalty parameter. Such problem can be addressed by iteratively solving the following subproblems for $\mathbf{x}$ and $\mathbf{z}$ while keeping the rest of the variables fixed

$$\begin{cases} \mathbf{x}_k = \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathcal{T}(\mathbf{x})\|^2 + \mu\sigma^2\|\mathbf{x} - \mathbf{z}_{k-1}\|^2 \\ \mathbf{z}_k = \arg\min_{\mathbf{z}} \frac{1}{2(\sqrt{\lambda/\mu})^2}\|\mathbf{z} - \mathbf{x}_k\|^2 + \mathcal{R}(\mathbf{z}). \end{cases} \tag{6}$$

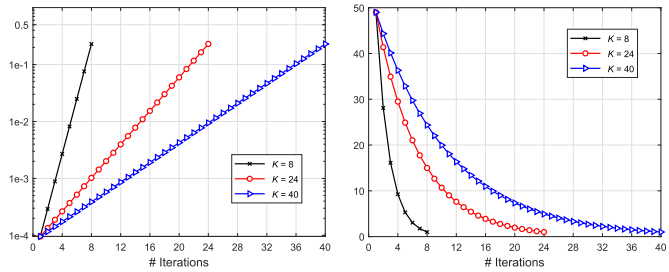
As such the data term and prior term are decoupled into two separate subproblems. To be specific, the subproblem of (6a) aims to find a proximal point of $\mathbf{z}_{k-1}$ and usually has a fast closed-form solution depending on $\mathcal{T}$, while the subproblem of (6b), from a Bayesian perspective, corresponds to Gaussian denoising on $\mathbf{x}_k$ with noise level $\sqrt{\lambda/\mu}$. Consequently, any Gaussian denoiser can be plugged into the alternating iterations to solve (2). To address this, we rewrite (6b) as follows:

$$\mathbf{z}_k = Denoiser(\mathbf{x}_k, \sqrt{\lambda/\mu}). \tag{7}$$

One can have two observations from (7). First, the prior $\mathcal{R}(\cdot)$ can be implicitly specified by a denoiser. For this reason, both the prior and the denoiser for plug-and-play IR are usually termed as denoiser prior. Second, it is interesting to learn a single CNN denoiser to replace (7) so as to exploit the advantages of CNN, such as high flexibility of network design, high efficiency on GPUs and powerful modeling capacity with deep networks.

4.2 General Methodology for Parameter Setting

From the alternating iterations between (6a) and (6b), it is easy to see that there involves three adjustable parameters, including penalty parameter μ, regularization parameter λ and the total number of iterations K.



(a) α_k (b) σ_k

Fig. 5. The values of α_k and σ_k at kth iteration with respect to different number of iterations $K = 8$, 24, and 40.

To guarantee $\mathbf{x}_k$ and $\mathbf{z}_k$ converge to a fixed point, a large μ is needed, which however requires a large K for convergence. Hence, the common way is to adopt the continuation strategy to gradually increase μ, resulting in a sequence of $\mu_1 < \cdots < \mu_k < \cdots < \mu_K$. Nevertheless, a new parameter needs to be introduced to control the step size, making the parameter setting more complicated. According to (7), we can observe that μ controls the noise level $\sigma_k (\triangleq \sqrt{\lambda/\mu_k})$ in kth iteration of the denoiser prior. On the other hand, a noise level range of $[0, 50]$ is supposed to be enough for σ_k. Inspired by such domain knowledge, we can instead set σ_k and λ to implicitly determine μ_k. Based on the fact that μ_k should be monotonically increasing, we uniformly sample σ_k from a large noise level σ_1 to a small one σ_K in log space. This means that μ_k can be easily determined via $\mu_k = \lambda/\sigma_k^2$. Following [17], σ_1 is fixed to 49 while σ_K is determined by the image noise level σ. Since K is user-specified and σ_K has clear physical meanings, they are practically easy to set. As for the theoretical convergence of plug-and-play IR, please refer to [31].

By far, the remaining parameter for setting is λ. Due to the fact that λ comes from the prior term and thus should be fixed, we can choose the optimal λ by a grid search on a validation dataset. Empirically, λ can yield favorable performance from the range of $[0.19, 0.55]$. In this paper, we fix it to 0.23 unless otherwise specified. It should be noted that since λ can be absorbed into σ and plays the role of controlling the trade-off between data term and prior term, one can implicitly tune λ by multiplying σ by a scalar. To have a clear understanding of the parameter setting, by denoting $\alpha_k \triangleq \mu_k \sigma^2 = \lambda\sigma^2/\sigma_k^2$ and assuming $\sigma_K = \sigma = 1$, we plot the values of α_k and σ_k with respect to different number of iterations $K = 8$, 24, and 40 in Fig. 5.

4.3 Periodical Geometric Self-Ensemble

Geometric self-ensemble based on flipping and rotation is a commonly-used strategy to boost IR performance [75]. It first transforms the input via flipping and rotation to generate 8 images, then gets the corresponding restored images after feeding the model with the 8 images, and finally produces the averaged result after the inverse transformation. While a performance gain can be obtained via geometric self-ensemble, it comes at the cost of increased inference time.

Different from the above method, we instead periodically apply the geometric self-ensemble for every successive 8 iterations. In each iteration, there involves one transformation before denoising and the counterpart inverse transformation after denoising. Note that the averaging step is abandoned because the input of the denoiser prior model varies across iterations. We refer to this method as periodical geometric self-ensemble. Its distinct advantage is that the total inference time would not increase. We empirically found that geometric self-ensemble can generally improve the PSNR by 0.02dB$\sim$0.2dB.

Based on the above discussion, we summarized the detailed algorithm of HQS-based plug-and-play IR with deep denoiser prior, namely DPIR, in Algorithm 1.

Algorithm 1. Plug-and-Play Image Restoration With Deep Denoiser Prior (DPIR)

Input : Deep denoiser prior model, degraded image $\mathbf{y}$, degradation operation $\mathcal{T}$, image noise level σ, σ_k of denoiser prior model at kth iteration for a total of K iterations, trade-off parameter λ.

Output : Restored image $\mathbf{z}_K$.

1: Initialize $\mathbf{z}_0$ from $\mathbf{y}$, pre-calculate $\alpha_k \triangleq \lambda\sigma^2/\sigma_k^2$.
2: **for** $k = 1, 2, \ldots, K$ **do**
3: $\quad \mathbf{x}_k = \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathcal{T}(\mathbf{x})\|^2 + \alpha_k \|\mathbf{x} - \mathbf{z}_{k-1}\|^2$; // *Solving data subproblem*
4: $\quad \mathbf{z}_k = Denoiser(\mathbf{x}_k, \sigma_k)$; // *Denoising with deep DRUNet denoiser and periodical geometric self-ensemble*
5: **end**

5 Experiments

To validate the effectiveness of the proposed DPIR, we consider three classical IR tasks, including image deblurring, single image super-resolution (SISR), and color image demosaicing. For each task, we will provide the specific degradation model, fast solution of (6a) in Algorithm 1, parameter setting for K and σ_K, initialization of $\mathbf{z}_0$, and the performance comparison with other state-of-the-art methods. To further analyze DPIR, we also provide the visual results of $\mathbf{x}_k$ and $\mathbf{z}_k$ at intermediate iterations as well as the convergence curves. Note that in order to show the advantage of the powerful DRUNet denoiser prior over IRCNN denoiser prior, we refer to DPIR with IRCNN denoiser prior as IRCNN+.

5.1 Image Deblurring

The degradation model for deblurring a blurry image with uniform blur (or image deconvolution) is generally expressed as

$$\mathbf{y} = \mathbf{x} \otimes \mathbf{k} + \mathbf{n}, \tag{8}$$

where $\mathbf{x} \otimes \mathbf{k}$ denotes two-dimensional convolution between the latent clean image $\mathbf{x}$ and the blur kernel $\mathbf{k}$. By assuming the convolution is carried out with circular boundary conditions, the fast solution of (6a) is given by

$$\mathbf{x}_k = \mathcal{F}^{-1}\left(\frac{\overline{\mathcal{F}(\mathbf{k})}\mathcal{F}(\mathbf{y}) + \alpha_k \mathcal{F}(\mathbf{z}_{k-1})}{\overline{\mathcal{F}(\mathbf{k})}\mathcal{F}(\mathbf{k}) + \alpha_k}\right), \tag{9}$$

where the $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$ denote Fast Fourier Transform (FFT) and inverse FFT, $\overline{\mathcal{F}(\cdot)}$ denotes complex conjugate of

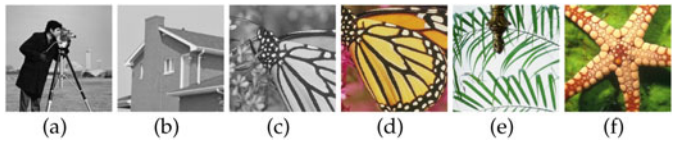


Fig. 6. Six classical testing images. (a) *Cameraman*; (b) *House*; (c) *Monarch*; (d) *Butterfly*; (e) *Leaves*; (f) *Starfish*.

$\mathcal{F}(\cdot)$. It can be noted that the blur kernel $\mathbf{k}$ is only involved in (9). In other words, (9) explicitly handles the distortion of blur.

5.1.1 Quantitative and Qualitative Comparison

For the sake of making a quantitative analysis on the proposed DPIR, we consider six classical testing images as shown in Fig. 6 and two of the eight real blur kernels from [76]. Specifically, the testing images which we refer to as Set6 consist of 3 grayscale images and 3 color images. Among them, *House* and *Leaves* are full of repetitive structures and thus can be used to evaluate non-local self-similarity prior. For the two blur kernels, they are of size 17×17 and 27×27, respectively. As shown in Table 6, we also consider Gaussian noise with different noise levels 2.55(1%) and 7.65(3%). Following the common setting, we synthesize the blurry images by first applying a blur kernel and then adding AWGN with noise level σ. For the parameters K and σ_K, they are set to 8 and σ, respectively. For $\mathbf{z}_0$, it is initialized as $\mathbf{y}$.

To evaluate the effectiveness of the proposed DPIR, we choose four representative methods for comparison, including model-based method EPLL [4], learning-based non-blind method FDN [77], learning-based blind method DMPHN [78] and plug-and-play method IRCNN and IRCNN+. Table 6 summarizes the PSNR results on Set6. As one can see, DMPHN obtains the lowest PSNR values possibly due to the lacking of the FFT module. In contrast, the proposed DPIR outperforms EPLL and FDN by a large margin. Although DPIR has 8 iterations rather than 30 iterations of IRCNN, it has a PSNR gain of 0.2dB∼2dB over IRCNN. On the other hand, with the same number of iterations, DPIR significantly outperforms IRCNN+, which indicates that the denoiser plays a vital role in plug-and-play IR. In addition, one can see that the PSNR gain of DPIR over IRCNN and IRCNN+ on *House* and *Leaves* is larger than those on other images. A possible reason is that the DRUNet denoiser learns more nonlocal self-similarity prior than the shallow denoisers of IRCNN.

The visual comparison of different methods on *Leaves* with the fourth kernel and noise level 7.65 is shown in Fig. 7. We can see that DMPHN can remove the noise but fail to recover the image sharpness, while FDN tends to smooth out fine details and generate color artifacts. Although IRCNN and IRCNN+ avoid the color artifacts, it fails to recover the fine details. In contrast, the proposed DPIR can recover image sharpness and naturalness.

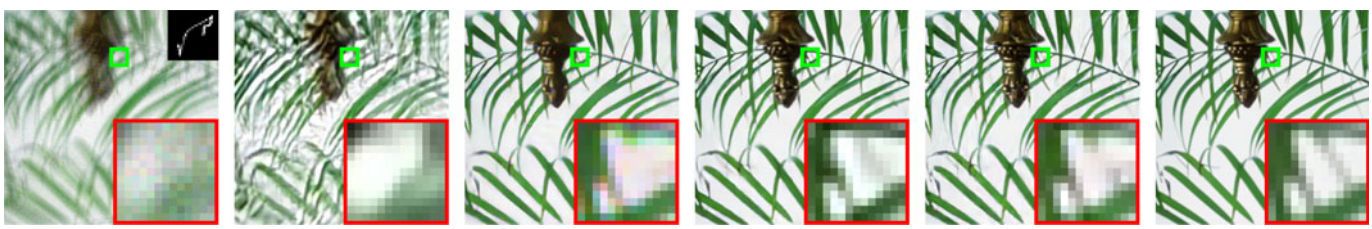
TABLE 6
PSNR(dB) Results of Different Methods on Set6 for Image Deblurring

Methods	σ	*C.man*	*House*	*Monarch*	*Butterfly*	*Leaves*	*Starfish*
The second kernel of size 17×17 from [76]							
EPLL	2.55	29.18	32.33	27.32	24.96	23.48	28.05
FDN	2.55	29.09	29.75	29.13	28.06	27.04	28.12
DMPHN	2.55	21.35	21.85	17.87	18.39	16.20	20.33
IRCNN	2.55	31.69	35.04	32.71	33.13	33.51	33.15
IRCNN+	2.55	31.23	34.01	31.85	32.55	32.66	32.34
DPIR	2.55	32.05	35.82	33.38	34.26	35.19	34.21
EPLL	7.65	24.82	28.50	23.03	20.82	20.06	24.23
FDN	7.65	26.18	28.01	25.86	24.76	23.91	25.21
DMPHN	7.65	20.71	22.45	19.01	17.97	15.83	19.74
IRCNN	7.65	27.70	31.94	28.23	28.73	28.63	28.76
IRCNN+	7.65	27.64	31.00	27.66	28.52	28.17	28.50
DPIR	7.65	28.17	32.79	28.48	29.52	30.11	29.83
The fourth kernel of size 27×27 from [76]							
EPLL	2.55	27.85	28.13	22.92	20.55	19.22	24.84
FDN	2.55	28.78	29.29	28.60	27.40	26.51	27.48
DMPHN	2.55	16.28	17.14	12.94	11.90	9.85	17.32
IRCNN	2.55	31.56	34.73	32.42	32.74	33.22	32.53
IRCNN+	2.55	31.29	34.17	31.82	32.48	33.59	32.18
DPIR	2.55	31.97	35.52	32.99	34.18	35.12	33.91
EPLL	7.65	24.31	26.02	20.86	18.64	17.54	22.47
FDN	7.65	26.13	27.41	25.39	24.27	23.53	24.71
DMPHN	7.65	15.50	16.63	12.51	11.13	9.74	15.07
IRCNN	7.65	27.58	31.55	27.99	28.53	28.45	28.42
IRCNN+	7.65	27.49	30.80	27.54	28.40	28.14	28.20
DPIR	7.65	27.99	32.87	28.27	29.45	30.27	29.46

The best and second best results are highlighted in red and blue colors, respectively.

5.1.2 Intermediate Results and Convergence

Figs. 8a, 8b, 8c, 8d, and 8e provide the visual results of $\mathbf{x}_k$ and $\mathbf{z}_k$ at different iterations on the testing image from Fig. 7, while Fig. 8f shows the PSNR convergence curves for $\mathbf{x}_k$ and $\mathbf{z}_k$. We can have the following observations. First, while (6a) can handle the distortion of blur, it also aggravates the strength of noise compared to its input $\mathbf{z}_{k-1}$. Second, the deep denoiser prior plays the role of removing noise, leading to a noise-free $\mathbf{z}_k$. Third, compared with $\mathbf{x}_1$ and $\mathbf{x}_2$, $\mathbf{x}_8$ contains more fine details, which means (6a) can iteratively recover the details. Fourth, according to Fig. 8f, $\mathbf{x}_k$ and $\mathbf{z}_k$ enjoy a fast convergence to the fixed point.



(a) Blurry image (b) DMPHN (9.74dB) (c) FDN (23.53dB) (d) IRCNN (28.45dB) (e) IRCNN+ (28.14dB) (f) DPIR (30.27dB)

Fig. 7. Visual results comparison of different deblurring methods on *Leaves*. The blur kernel is visualized in the upper right corner of the blurry image. The noise level is 7.65(3%).

(a) $\mathbf{x}_1$ (16.34dB) (b) $\mathbf{z}_1$ (23.75dB) (c) $\mathbf{x}_4$ (22.33dB) (d) $\mathbf{z}_4$ (29.37dB) (e) $\mathbf{x}_8$ (29.34dB) (f) Convergence curves

Fig. 8. (a)-(e) Visual results and PSNR results of $\mathbf{x}_k$ and $\mathbf{z}_k$ at different iterations; (f) Convergence curves of PSNR (y-axis) for $\mathbf{x}_k$ and $\mathbf{z}_k$ with respect to number of iterations (x-axis).

5.1.3 Analysis of the Parameter Setting

While we fixed the total number of iterations K to be 8 and the noise level in the first iteration σ_1 to be 49, it is interesting to investigate the performance with other settings. Table 7 reports the PSNR results with different combinations of K and σ_1 on the testing image from Fig. 7. One can see that larger σ_1, such as 39 and 49, could result in better PSNR results. On the other hand, if σ_1 is small, a large K needs to be specified for a good performance, which however would increase the computational burden. As a result, K and σ_1 play an important role for the trade-off between efficiency and effectiveness.

5.1.4 Results of DPIR With Blind DRUNet Denoiser

It is interesting to demonstrate the performance of DPIR with blind DRUNet denoiser. For this purpose, we have trained a blind DRUNet denoiser by removing the noise level map. With the same parameter setting, we provide the visual results and PSNR results of $\mathbf{x}_k$ and $\mathbf{z}_k$ at different iterations of DPIR with the blind DRUNet denoiser on the noisy and blurry *Leaves* in Fig. 9. Note that the testing image is same as in Figs. 7 and 8. By comparison, we can see that DPIR with blind denoiser gives rise to much lower PSNR values than DPIR with non-blind denoiser. In particular, some structural artifacts can be observed with a closer look at the final result $\mathbf{z}_8$. As a result, the non-blind denoiser is more suitable than blind denoiser for plug-and-play IR.

5.2 Single Image Super-Resolution (SISR)

While existing SISR methods are mainly designed for bicubic degradation model with the following formulation:

$$\mathbf{y} = \mathbf{x} \downarrow_s^{bicubic}, \tag{10}$$

where $\downarrow_s^{bicubic}$ denotes bicubic downsamling with downscaling factor s, it has been revealed that these methods would deteriorate seriously if the real degradation model deviates from the assumed one [79], [80]. To remedy this, an alternative way is to adopt a classical but practical degradation model which assumes the low-resolution (LR) image is a blurred, decimated, and noisy version of high-resolution (HR) image. The mathematical formulation of such degradation model is given by

$$\mathbf{y} = (\mathbf{x} \otimes \mathbf{k}) \downarrow_s + \mathbf{n}, \tag{11}$$

where $\downarrow_s$ denotes the standard s-fold downsampler, i.e., selecting the upper-left pixel for each distinct $s \times s$ patch.

In this paper, we consider the above-mentioned two degradation models for SISR. As for the solution of (6a), the following iterative back-projection (IBP) solution [25], [81] can be adopted for bicubic degradation

$$\mathbf{x}_k = \mathbf{z}_{k-1} - \gamma(\mathbf{y} - \mathbf{z}_{k-1} \downarrow_s^{bicubic}) \uparrow_s^{bicubic}, \tag{12}$$

where $\uparrow_s^{bicubic}$ denotes bicubic interpolation with upscaling factor s, γ is the step size. Note that we only show one iteration for simplicity. As an extension, (12) can be further modified as follows to handle the classical degradation model

$$\mathbf{x}_k = \mathbf{z}_{k-1} - \gamma\Big((\mathbf{y} - (\mathbf{z}_{k-1} \otimes \mathbf{k}) \downarrow_s) \uparrow_s \Big) \otimes \mathbf{k}, \tag{13}$$

where $\uparrow_s$ denotes upsampling the spatial size by filling the new entries with zeros. Especially noteworthy is that there exists a fast close-form solution to replace the above iterative scheme. According to [82], by assuming the convolution is carried out with circular boundary conditions as in deblurring, the closed-form solution is given by

$$\mathbf{x}_k = \mathcal{F}^{-1}\left(\frac{1}{\alpha_k}\left(\mathbf{d} - \overline{\mathcal{F}(\mathbf{k})} \odot_s \frac{(\mathcal{F}(\mathbf{k})\mathbf{d}) \Downarrow_s}{(\overline{\mathcal{F}(\mathbf{k})}\mathcal{F}(\mathbf{k})) \Downarrow_s + \alpha_k}\right)\right), \tag{14}$$

where $\mathbf{d} = \overline{\mathcal{F}(\mathbf{k})}\mathcal{F}(\mathbf{y} \uparrow_s) + \alpha_k \mathcal{F}(\mathbf{z}_{k-1})$ and where $\odot_s$ denotes distinct block processing operator with element-wise multi-

TABLE 7
PSNR Results With Different Combinations of K and σ_1 on the Testing Image From Fig. 7

K	$\sigma_1 = 9$	$\sigma_1 = 19$	$\sigma_1 = 29$	$\sigma_1 = 39$	$\sigma_1 = 49$
4	20.04	23.27	25.70	27.65	28.96
8	22.50	25.96	28.40	29.89	30.27
24	26.58	29.64	30.06	30.13	30.16
40	28.60	29.82	29.92	29.98	30.01

(a) $\mathbf{x}_4$ (21.66dB) (b) $\mathbf{z}_4$ (27.64dB) (c) $\mathbf{z}_8$ (25.59dB)

Fig. 9. Visual results and PSNR results of $\mathbf{x}_k$ and $\mathbf{z}_k$ at different iterations of the proposed DPIR with blind DRUNet denoiser. The testing image is same as in Figs. 7 and 8.

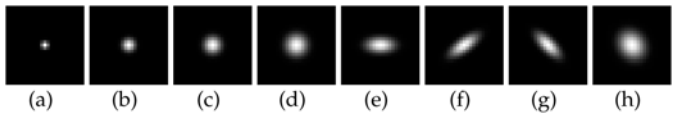


Fig. 10. The eight testing Gaussian kernels for SISR. (a)-(d) are isotropic Gaussian kernels; (e)-(f) are anisotropic Gaussian kernels.

plication, i.e., applying element-wise multiplication to the $s \times s$ distinct blocks of $\overline{\mathcal{F}(\mathbf{k})}$, $\Downarrow_s$ denotes distinct block downsampler, i.e., averaging the $s \times s$ distinct blocks [83]. It is easy to verify that (9) is a special case of (14) with $s = 1$. It is worth noting that (11) can also be used to solve bicubic degradation by setting the blur kernel to the approximated bicubic kernel [83]. In general, the closed-form solution (14) should be advantageous over iterative solutions (13). The reason is that the former is an exact solution which contains one parameter (i.e., α_k) whereas the latter is an inexact solution which involves two parameters (i.e., number of inner iterations per outer iteration and step size).

For the overall parameter setting, K and σ_K are set to 24 and $\max(\sigma, s)$, respectively. For the parameters in (12) and (13), γ is fixed to 1.75, the the number of inner iterations required per outer iteration is set to 5. For the initialization of z_0, the bicubic interpolation of the LR image is utilized. In particular, since the classical degradation model selects the upper-left pixel for each distinct $s \times s$ patch, a shift problem should be properly addressed. To tackle with this, we adjust z_0 by using 2D linear grid interpolation.

5.2.1 Quantitative and Qualitative Comparison

To evaluate the flexibility of DPIR, we consider bicubic degradation model, and classical degradation model with 8 diverse Gaussian blur kernels as shown in Fig. 10. Following [83], the 8 kernels consist of 4 isotropic kernels with different standard deviations (i.e., 0.7, 1.2, 1.6 and 2.0) and 4 anisotropic kernels. We do not consider motion blur kernels since it has been pointed out that Gaussian kernels are enough for SISR task. To further analyze the performance, three different combinations of scale factor and noise level, including ($s = 2$, $\sigma = 0$), ($s = 3$, $\sigma = 0$) and ($s = 3$, $\sigma = 7.65$), are considered.

For the compared methods, we consider the bicubic interpolation method, RCAN [84], SRFBN [85], MZSR [86], IRCNN and IRCNN+. Specifically, RCAN is the state-of-the-art bicubic degradation based deep model consisting of about 400 layers. SRFBN is a recurrent neural network with feed-back mechanism. Note that we do not retrain the RCAN and SRFBN models to handle the testing degradation cases as they lack flexibility. Moreover, it is unfair because our DPIR can handle a much wider range of degradations. MZSR is a zero-shot method based on meta-transfer learning which learns an initial network and then fine-tunes the model on a pair of given LR image and its re-degraded LR image with a few gradient updates. Similar to IRCNN and DPIR, MZSR is a non-blind method that assumes the blur kernel is known beforehand. Since MZSR needs to downsample the LR image for fine-tuning, the scale factor should be not too large in order to capture enough information. As a result, MZSR is mainly designed for scale factor 2.

Table 8 reports the average PSNR(dB) results of different methods for bicubic degradation and classical degradation on color BSD68 dataset. From Table 8, we can have several observations. First, as expected, RCAN and SRFBN achieve promising results on bicubic degradation with $\sigma = 0$ but lose effectiveness when the true degradation deviates from the assumed one. We note that SAN [87] has a similar performance to RCAN and SRFBN since they are trained for bicubic degradation. Second, with the accurate classical degradation model, MZSR outperforms RCAN on most of the blur kernels. Third, IRCNN has a clear PSNR gain over MZSR on smoothed blur kernel. The reason is that MZSR relies heavily on the internal learning of LR image. Fourth, IRCNN performs better on bicubic kernel and the first isotropic Gaussian kernel with noise level $\sigma = 0$ than others. This indicates that the IBP solution has very limited generalizability. On the other hand, IRCNN+ has a much higher PSNR than IRCNN, which demonstrates the advantage of closed-form solution over the IBP solution. Last, DPIR can further improves over IRCNN+ by using a more powerful denoiser.

Fig. 11 shows the visual comparison of different SISR methods on an image corrupted by classical degradation model. It can be observed that MZSR and IRCNN produce better visual results than bicubic interpolation method. With an inaccurate data term solution, IRCNN fails to recover sharp edges. In comparison, by using a closed-form data term solution, IRCNN+ can produce much better results with sharp edges. Nevertheless, it lacks the ability to recover clean HR image. In contrast, with a strong denoiser prior, DPIR produces the best visual result with both sharpness and naturalness.

TABLE 8
Average PSNR(dB) Results of Different Methods for Single Image Super-Resolution on CBSD68 Dataset

Methods	Bicubic	RCAN	SRFBN	MZSR	IRCNN	IRCNN+	DPIR	Bicubic	RCAN	SRFBN	IRCNN	IRCNN+	DPIR	Bicubic	RCAN	SRFBN	IRCNN	IRCNN+	DPIR
Kernel	$s = 2$, $\sigma = 0$							$s = 3$, $\sigma = 0$						$s = 3$, $\sigma = 7.65(3\%)$					
(a)	27.60	29.50	29.39	28.89	29.92	30.00	29.78	25.83	25.02	24.98	26.43	26.56	26.50	24.65	22.77	22.58	25.45	25.58	26.01
(b)	26.14	26.77	26.75	29.45	29.49	30.28	30.16	25.57	27.37	27.31	26.88	27.12	27.08	24.45	24.01	24.04	25.28	25.30	26.10
(c)	25.12	25.32	25.31	28.49	27.75	29.23	29.72	24.92	25.87	25.84	26.56	27.23	27.21	23.94	23.42	23.45	24.61	24.92	25.71
(d)	24.31	24.37	24.35	25.26	26.44	27.82	28.71	24.27	24.69	24.70	25.78	27.08	27.18	23.41	22.76	22.79	23.97	24.63	25.17
(e)	24.29	24.38	24.37	25.48	26.41	27.76	28.40	24.20	24.65	24.64	25.55	26.78	27.05	23.35	22.71	22.73	23.96	24.58	25.09
(f)	24.02	24.10	24.09	25.46	26.05	27.72	28.50	23.98	24.46	24.43	25.44	26.87	27.04	23.16	22.54	22.55	23.75	24.51	25.01
(g)	24.16	24.24	24.22	25.93	26.28	27.86	28.66	24.10	24.63	24.61	25.64	27.00	27.11	23.27	22.64	22.68	23.87	24.59	25.12
(h)	23.61	23.61	23.59	22.27	25.45	26.88	27.57	23.63	23.82	23.82	24.92	26.55	26.94	22.88	22.18	22.20	23.41	24.27	24.60
Bicubic	26.37	31.18	31.07	29.47	30.31	30.34	30.12	25.97	28.08	28.00	27.19	27.24	27.23	24.76	24.21	24.24	24.36	25.61	26.35

The best and second best results are highlighted in red and blue colors, respectively.

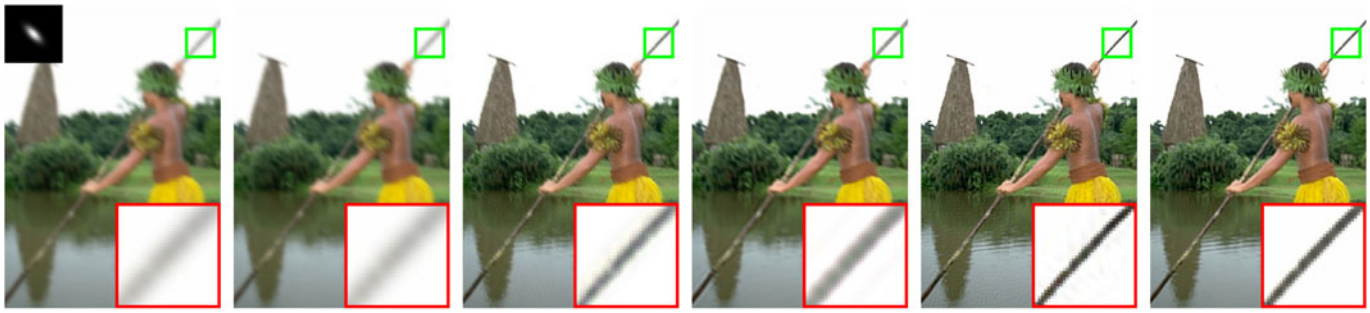


Fig. 11. Visual results comparison of different SISR methods on an image corrupted by classical degradation model. The kernel is shown on the upper-left corner of the bicubicly interpolated LR image. The scale factor is 2.

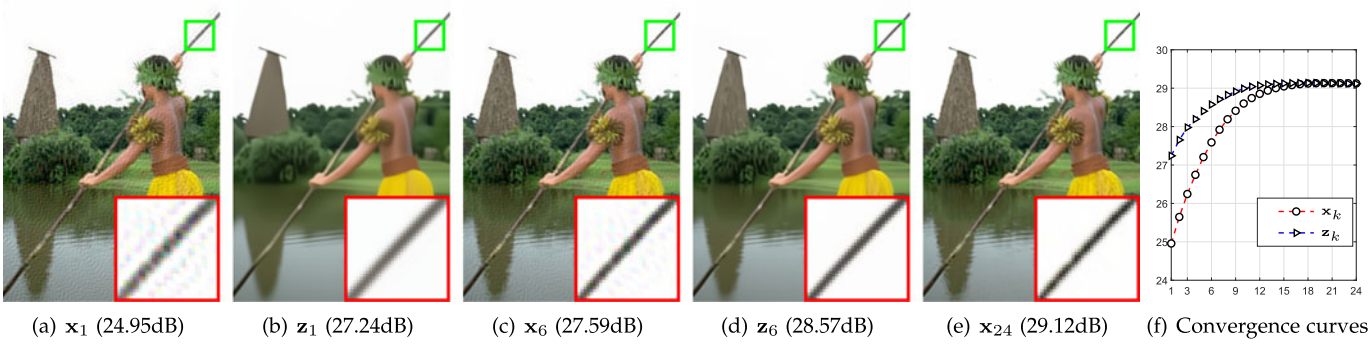


Fig. 12. (a)-(e) Visual results and PSNR results of $\mathbf{x}_k$ and $\mathbf{z}_k$ at different iteration; (f) Convergence curves of PSNR results (y-axis) for $\mathbf{x}_k$ and $\mathbf{z}_k$ with respect to number of iterations (x-axis).

5.2.2 Intermediate Results and Convergence

Figs. 12a, 12b, 12c, 12d, and 12e provides the visual results and PSNR results of $\mathbf{x}_k$ and $\mathbf{z}_k$ at different iterations of DPIR on the testing image from Fig. 11. One can observe that, although the LR image contains no noise, the the closed-form solution $\mathbf{x}_1$ would introduce severe structured noise. However, it has a better PSNR than that of RCAN. After passing $\mathbf{x}_1$ through the DRUNet denoiser, such structured noise is removed as can be seen from $\mathbf{z}_1$. Meanwhile, the tiny textures and structures are smoothed out and the edges become blurry. Nevertheless, the PSNR is significantly improved and is comparable to that of MZSR. As the number of iterations increases, $\mathbf{x}_6$ contains less structured noise than $\mathbf{x}_1$, while $\mathbf{z}_6$ recovers more details and sharper edges than $\mathbf{z}_1$. The corresponding PSNR convergence curves are plotted in Fig. 12f, from which we can see that $\mathbf{x}_k$ and $\mathbf{z}_k$ converge quickly to the fixed point.

5.3 Color Image Demosaicing

Current consumer digital cameras mostly use a single sensor with a color filter array (CFA) to record one of three R, G, and B values at each pixel location. As an essential process in camera pipeline, demosaicing aims to estimate the missing pixel values from a one-channel mosaiced image and the corresponding CFA pattern to recover a three-channel image [88], [89], [90]. The degradation model of mosaiced image can be expressed as

$$\mathbf{y} = \mathbf{M} \odot \mathbf{x}, \tag{15}$$

where $\mathbf{M}$ is determined by CFA pattern and is a matrix with binary elements indicating the missing pixels of $\mathbf{y}$, and $\odot$ denotes element-wise multiplication. The closed-from solution of (6a) is given by

$$\mathbf{x}_{k+1} = \frac{\mathbf{M} \odot \mathbf{y} + \alpha_k \mathbf{z}_k}{\mathbf{M} + \alpha_k}. \tag{16}$$

In this paper, we consider the commonly-used Bayer CFA pattern with RGGB arrangement. For the parameters K and σ_K, they are set to 40 and 0.6, respectively. For z_0, it is initialized by matlab's `demosaic` function.

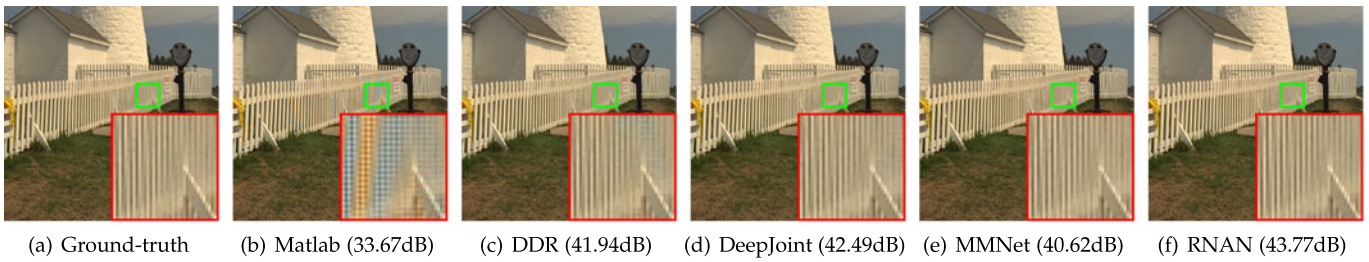
5.3.1 Quantitative and Qualitative Comparison

To evaluate the performance of DPIR for color image demosaicing, the widely-used Kodak dataset (consisting of 24 color images of size 768×512) and McMaster dataset (consisting of 18 color images of size 500×500) are used. The corresponding mosaiced images are obtained by filtering the color images with the Bayer CFA pattern. The compared methods include matlab's `demosaic` function [88], directional difference regression (DDR) [91], deep unfolding majorization-minimization network (MMNet) [92], residual learning-based joint demosaicing-denoising (RLDD) [93], deep joint demosaicing and denoising (DeepJoint) [89], very deep residual non-local attention network (RNAN) [58] learned simultaneous sparse coding (LSSC) [94], iterative residual interpolation (IRI) [95], minimized-Laplacian residual interpolation (MLRI) [96], primal-dual algorithm with CBM3D denoiser prior (FlexISP) [15], IRCNN and IRCNN+. Note that DDR, MMNet, RLDD, DeepJoint, and RNAN are learning-based methods, while LSSC, IRI, MLRI, FlexISP, IRCNN, IRCNN+, and DPIR are model-based methods.

TABLE 9
Demosaicing Results of Different Methods on Kodak and McMaster Datasets

Datasets	Matlab	DDR	DeepJoint	MMNet	RLDD	RNAN	LSSC	IRI	FlexISP	IRCNN	IRCNN+	DPIR
Kodak	35.78	41.11	42.00	40.19	42.49	43.16	41.43	39.23	38.52	40.29	40.80	42.68
McMaster	34.43	37.12	39.14	37.09	39.25	39.70	36.15	36.90	36.87	37.45	37.79	39.39

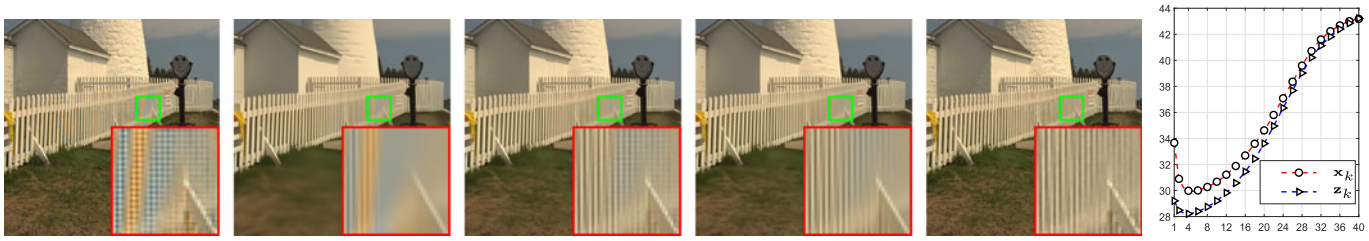
The best and second best results are highlighted in red and blue colors, respectively.



(a) Ground-truth (b) Matlab (33.67dB) (c) DDR (41.94dB) (d) DeepJoint (42.49dB) (e) MMNet (40.62dB) (f) RNAN (43.77dB)

(g) LSSC (42.31dB) (h) IRI (39.49dB) (i) FlexISP (36.95dB) (j) IRCNN (40.18dB) (k) IRCNN+ (40.85dB) (l) DPIR (43.23dB)

Fig. 13. Visual results comparison of different demosaicing methods on image *kodim19* from Kodak dataset.



(a) $\mathbf{x}_1$ (33.67dB) (b) $\mathbf{z}_1$ (29.21dB) (c) $\mathbf{x}_{16}$ (32.69dB) (d) $\mathbf{z}_{16}$ (31.45dB) (e) $\mathbf{x}_{40}$ (43.18dB) (f) Convergence curves

Fig. 14. (a)-(e) Visual results and PSNR results of $\mathbf{x}_k$ and $\mathbf{z}_k$ at different iterations; (f) Convergence curves of PSNR results (y-axis) for $\mathbf{x}_k$ and $\mathbf{z}_k$ with respect to number of iterations (x-axis).

Table 9 reports the average PSNR(dB) results of different methods on Kodak dataset and McMaster dataset. It can be seen that while RNAN and MMNet achieve the best results, DPIR can have a very similar result and significantly outperforms the other model-based methods. With a stronger denoiser, DPIR has an average PSNR improvement up to 1.8dB over IRCNN+.

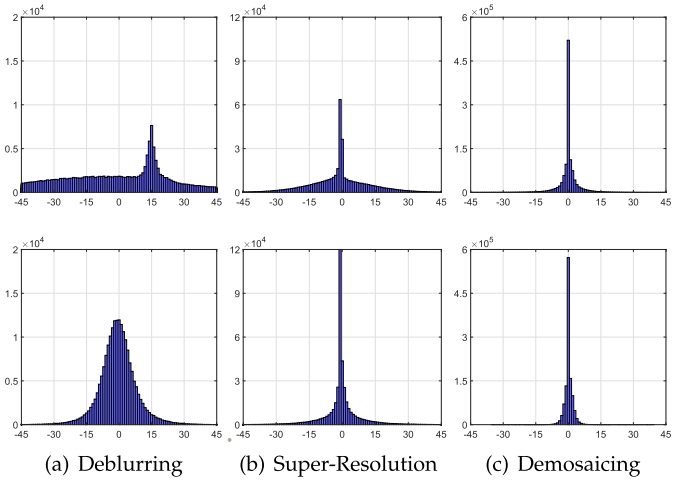
Fig. 13 shows the visual results comparison of different methods on a testing image from Kodak dataset. As one can see, the Matlab's simple demosaicing method introduces some zipper effects and false color artifacts. Such artifacts are highly reduced by learning-based methods such as DeepJoint, MMNet and RNAN. For the model-based methods, DPIR produces the best visual results whereas the others give rise to noticeable artifacts.

5.3.2 Intermediate Results and Convergence

Figs. 14a, 14b, 14c, 14d, and 14e show the visual results and PSNR results of $\mathbf{x}_k$ and $\mathbf{z}_k$ at different iterations. One can see that the DRUNet denoiser prior plays the role of smoothing out current estimation $\mathbf{x}$. By passing $\mathbf{z}$ through (16), the new output $\mathbf{x}$ obtained by a weighted average of $\mathbf{y}$ and $\mathbf{z}$ becomes unsmooth. In this sense, the denoiser also aims to diffuse $\mathbf{y}$ for a better estimation of missing values. Fig. 14f shows the PSNR convergence curves of $\mathbf{x}_k$ and $\mathbf{z}_k$. One can see that the two PSNR sequences are not monotonic but they eventually converge to the fixed point. Specifically, a decrease of the PSNR value for the first four iterations can be observed as the denoiser with a large noise level removes much more useful information than the unwanted artifacts.

6 Discussion

While the denoiser prior for plug-and-play IR is trained for Gaussian denoising, it does not necessary mean the noise of its input (or more precisely, the difference to the ground-truth) has a Gaussian distribution. In fact, the noise distribution varies across different IR tasks and even different iterations. Fig. 15 shows the noise histogram of $\mathbf{x}_1$ and $\mathbf{x}_8$ in



(a) Deblurring (b) Super-Resolution (c) Demosaicing

Fig. 15. Histogram of the noise (difference) between the ground-truth and input of the denoiser in the first iteration (first row) and last iteration (second row) for (a) deblurring, (b) super-resolution, and (c) demosaicing. The histograms are based on $\mathbf{x}_1$ and $\mathbf{x}_8$ in Fig. 8, $\mathbf{x}_1$ and $\mathbf{x}_{24}$ in Fig. 12 and $\mathbf{x}_1$ and $\mathbf{x}_{40}$ in Fig. 14.

Fig. 8 for deblurring, $\mathbf{x}_1$ and $\mathbf{x}_{24}$ in Fig. 12 for super-resolution, and $\mathbf{x}_1$ and $\mathbf{x}_{40}$ in Fig. 14 for demosaicing. It can be observed that the three IR tasks has very different noise distributions. This is intuitively reasonable because the noise also correlates with the degradation operation which is different for the three IR tasks. Another interesting observation is that the two noise distributions of $\mathbf{x}_1$ and $\mathbf{x}_8$ in Fig. 15a are different and the latter tends to be Gaussian-like. The underlying reason is that the blurriness caused by blur kernel is gradually alleviated after several iterations. In other words, $\mathbf{x}_8$ suffers much less from the blurriness and thus is dominated by Gaussian-like noise.

According to the experiments and analysis, it can be concluded that the denoiser prior mostly removes the noise along with some fine details, while the subsequent data subproblem plays the role of alleviating the noise-irrelevant degradation and adding the lost details back. Such mechanisms actually enable the plug-and-play IR to be a generic method. However, it is worth noting that this comes at the cost of losing efficiency and specialization because of such general-purpose Gaussian denoiser prior and the manual selection of hyper-parameters. In comparison, deep unfolding IR can train a compact inference with better performance by jointly learning task-specific denoiser prior and hyper-parameters. Taking SISR as an example, rather than smoothing out the fine details by deep plug-and-play denoiser, the deep unfolding denoiser can recover the high-frequency details.

7 Conclusion

In this paper, we have trained flexible and effective deep denoisers for plug-and-play image restoration. Specifically, by taking advantage of half-quadratic splitting algorithm, the iterative optimization of three different image restoration tasks, including deblurring, super-resolution and color image demosaicing, consists of alternately solving a data subproblem which has a closed-form solution and a prior subproblem which can be replaced by a deep denoiser. Extensive experiments and analysis on parameter setting, intermediate results, empirical convergence were provided. The results have demonstrated that plug-and-play image restoration with powerful deep denoiser prior have several advantages. On the one hand, it boosts the effectiveness of model-based methods due to the implicit but powerful prior modeling of deep denoiser. On the other hand, without task-specific training, it is more flexible than learning-based methods while having comparable performance. In summary, this work has highlighted the advantages of deep denoiser based plug-and-play image restoration. It is worth noting that there also remains room for further study. For example, one direction would be how to integrate other types of deep image prior, such as deep generative prior [97], for effective image restoration.

Acknowledgments

This work was supported in part by the ETH Zürich Fund OK, in part by Huawei, in part by Amazon AWS, in part by NVIDIA, and in part by the National Natural Science Foundation of China under Grant U19A2073.

References

[1] W. H. Richardson, "Bayesian-based iterative method of image restoration," *J. Opt. Soc. Amer.*, vol. 62, no. 1, pp. 55–59, 1972.
[2] H. C. Andrews and B. R. Hunt, "Digital image restoration, " in, *Prentice-Hall Signal Processing Series*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1977.
[3] S. Roth and M. J. Black, "Fields of experts," *Int. J. Comput. Vis.*, vol. 82, no. 2, pp. 205–229, 2009.
[4] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 479–486.
[5] M. F. Tappen, "Utilizing variational optimization to learn Markov random fields," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.
[6] A. Barbu, "Training an active random field for real-time image denoising," *IEEE Trans. Image Process.*, vol. 18, no. 11, pp. 2451–2462, Nov. 2009.
[7] J. Sun and M. F. Tappen, "Separable Markov random field model and its applications in low level vision," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 402–407, Jan. 2013.
[8] U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 2774–2781.
[9] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1256–1272, Jun. 2016.
[10] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 2862–2869.
[11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
[12] D. Geman and C. Yang, "Nonlinear image recovery with half-quadratic regularization," *IEEE Trans. Image Process.*, vol. 4, no. 7, pp. 932–946, Jul. 1995.
[13] N. Parikh and S. P. Boyd, "Proximal algorithms," *Foundations Trends Optim.*, vol. 1, no. 3, pp. 127–239, 2014.
[14] A. Danielyan, V. Katkovnik, and K. Egiazarian, "Image deblurring by augmented lagrangian with BM3D frame prior," in *Proc. Workshop Inf. Theor. Methods Sci. Eng.*, 2010, pp. 16–18.
[15] F. Heide *et al.*, "FlexISP: A flexible camera image processing framework," *ACM Trans. Graph.*, vol. 33, no. 6, 2014, Art. no. 231.
[16] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *Proc. IEEE Global Conf. Sig. Inf. Process.*, 2013, pp. 945–948.
[17] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3929–3938.

[18] K. Zhang, W. Zuo, and L. Zhang, "FFDNet: Toward a fast and flexible solution for CNN-based image denoising," *IEEE Trans. Image Process.*, vol. 27, no. 9, pp. 4608–4622, May 2018.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Int.*, 2015, pp. 234–241.

[21] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.

[22] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 60–65.

[23] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[24] A. Danielyan, V. Katkovnik, and K. Egiazarian, "BM3D frames and variational image deblurring," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1715–1728, Apr. 2012.

[25] K. Egiazarian and V. Katkovnik, "Single image super-resolution via BM3D sparse coding," in *Proc. Eur. Signal Process. Conf.*, 2015, pp. 2849–2853.

[26] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[27] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *J. Math. Imag. Vis.*, vol. 40, no. 1, pp. 120–145, 2011.

[28] A. M. Teodoro, J. M. Bioucas-Dias , and M. A. Figueiredo, "Image restoration and reconstruction using variable splitting and class-adapted image priors," in *Proc. IEEE Int. Conf. Image Process.*, 2016, pp. 3518–3522.

[29] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "From denoising to compressed sensing," *IEEE Trans. Inf. Theory*, vol. 62, no. 9, pp. 5117–5144, Sep. 2016.

[30] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of Gaussians in the wavelet domain," *IEEE Trans. Image Process.*, vol. 12, no. 11, pp. 1338–1351, Nov. 2003.

[31] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-play ADMM for image restoration: Fixed-point convergence and applications," *IEEE Trans. Comput. Imag.*, vol. 3, no. 1, pp. 84–98, Mar. 2016.

[32] U. S. Kamilov, H. Mansour, and B. Wohlberg, "A plug-and-play priors approach for solving nonlinear imaging inverse problems," *IEEE Signal Process. Lett.*, vol. 24, no. 12, pp. 1872–1876, Dec. 2017.

[33] Y. Sun, S. Xu, Y. Li, L. Tian, B. Wohlberg, and U. S. Kamilov, "Regularized fourier ptychography using an online plug-and-play algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 7665–7669.

[34] N. Yair and T. Michaeli, "Multi-scale weighted nuclear norm image restoration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3165–3174.

[35] R. G. Gavaskar and K. N. Chaudhury, "Plug-and-play ISTA converges with kernel denoisers," *IEEE Signal Process. Lett.*, vol. 27, pp. 610–614, Apr. 2020.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[37] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (red)," *SIAM J. Imag. Sci.*, vol. 10, no. 4, pp. 1804–1844, 2017.

[38] T. Tirer and R. Giryes, "Image restoration by iterative denoising and backward projections," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1220–1234, Mar. 2018.

[39] S. Gu, R. Timofte, and L. Van Gool , "Integrating local and non-local denoiser priors for image restoration," in *Proc. Int. Conf. Pattern Recognit.*, 2018, pp. 2923–2928.

[40] T. Tirer and R. Giryes, "Super-resolution via image-adapted denoising CNNs: Incorporating external and internal learning," *IEEE Signal Process. Lett.*, vol. 26, no. 7, pp. 1080–1084, Jul. 2019.

[41] Z. Li and J. Wu, "Learning deep CNN denoiser priors for depth image inpainting," *Appl. Sci.*, vol. 9, no. 6, 2019, Art. no. 1103.

[42] E. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, "Plug-and-play methods provably converge with properly trained denoisers," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5546–5557.

[43] Y. Sun, J. Liu, and U. Kamilov, "Block coordinate regularization by denoising," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 380–390.

[44] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.

[45] J. Zhang and B. Ghanem, "ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1828–1837.

[46] H. K. Aggarwal, M. P. Mani, and M. Jacob, "MoDL: Model-based deep learning architecture for inverse problems," *IEEE Trans. Med. Imag.*, vol. 38, no. 2, pp. 394–405, Feb. 2018.

[47] W. Dong, P. Wang, W. Yin, G. Shi, F. Wu, and X. Lu, "Denoising prior driven deep neural network for image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 10, pp. 2305–2318, Oct. 2018.

[48] C. Bertocchi, E. Chouzenoux, M.-C. Corbineau, J.-C. Pesquet, and M. Prato, "Deep unfolding of a proximal interior point method for image restoration," *Inverse Problems*, vol. 36, no. 3, 2020, Art. no. 034005.

[49] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2001, pp. 416–423.

[50] J. Lehtinen *et al.*, "Noise2Noise: Learning image restoration without clean data," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2965–2974.

[51] A. Krull, T.-O. Buchholz, and F. Jug, "Noise2Void-learning denoising from single noisy images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2129–2137.

[52] J. Batson and L. Royer, "Noise2Self: Blind denoising by self-supervision," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 524–533.

[53] S. Guo, Z. Yan, K. Zhang, W. Zuo, and L. Zhang, "Toward convolutional blind denoising of real photographs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1712–1722.

[54] T. Brooks, B. Mildenhall, T. Xue, J. Chen, D. Sharlet, and J. T. Barron, "Unprocessing images for learned raw denoising," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11036–11045.

[55] A. Abdelhamed, M. A. Brubaker, and M. S. Brown, "Noise flow: Noise modeling with conditional normalizing flows," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 3165–3173.

[56] S. W. Zamir *et al.*, "CycleISP: Real image restoration via improved data synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2696–2705.

[57] S. Lefkimmiatis, "Non-local color image denoising with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3587–3596.

[58] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, "Residual non-local attention networks for image restoration," in *Proc. Int. Conf. Learn. Representations*, 2019.

[59] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang, "Non-local recurrent network for image restoration," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1673–1682.

[60] T. Plötz and S. Roth, "Neural nearest neighbors networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1087–1098.

[61] Z. Zhang, Q. Liu, and Y. Wang, "Road extraction by deep residual U-Net," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 5, pp. 749–753, May 2018.

[62] G. Venkatesh, Y. Naresh, S. Little, and N. E. O'Connor,"A deep residual architecture for skin lesion segmentation," in *Proc. Context-Aware Oper. Theaters, Comput. Assist. Robot. Endoscopy, Clin. Image-Based Procedures, Skin Image Anal.*, 2018, pp. 277–284.

[63] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee , "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 136–144.

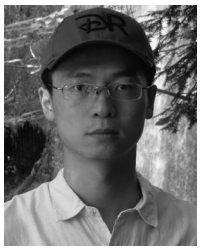
[64] S. Mohan, Z. Kadkhodaie, E. P. Simoncelli, and C. Fernandez-Granda , "Robust and interpretable blind image denoising via bias-free convolutional neural networks," in *Proc. Int. Conf. Learn. Representations*, 2019, *arXiv:1906.05478*.

[65] K. Ma *et al.*, "Waterloo exploration database: New challenges for image quality assessment models," *IEEE Trans. Image Process.*, vol. 26, no. 2, pp. 1004–1016, Feb. 2017.

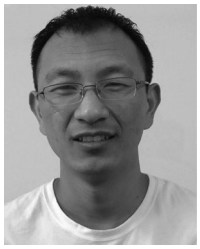
[66] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 126–135.

[67] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.
[68] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, 2006.
[69] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, "Residual non-local attention networks for image restoration," in *Proc. Int. Conf. Learn. Representations*, 2019.
[70] X. Jia, S. Liu, X. Feng, and L. Zhang, "FOCNet: A fractional optimal control network for image denoising," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 6054–6063.
[71] R. Franzen, "Kodak lossless true color image suite," vol. 4, no. 2, 1999.
[72] L. Zhang, X. Wu, A. Buades, and X. Li, "Color demosaicking by local directional interpolation and nonlocal adaptive thresholding," *J. Electron. Imag.*, vol. 20, no. 2, 2011, Art. no. 023016.
[73] C. Dong, Y. Deng, C. C. Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 576–584.
[74] M. Ehrlich, L. Davis, S.-N. Lim, and A. Shrivastava, "Quantization guided JPEG artifact correction," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp 293–309.
[75] R. Timofte, R. Rothe, and L. Van Gool , "Seven ways to improve example-based single image super resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1865–1873.
[76] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Understanding and evaluating blind deconvolution algorithms," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 1964–1971.
[77] J. Kruse, C. Rother, and U. Schmidt, "Learning to push the limits of efficient FFT-based image deconvolution," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 4586–4594.
[78] H. Zhang, Y. Dai, H. Li, and P. Koniusz, "Deep stacked hierarchical multi-patch network for image deblurring," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5978–5986.
[79] N. Efrat, D. Glasner, A. Apartsin, B. Nadler, and A. Levin, "Accurate blur models vs. image priors in single image super-resolution," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 2832–2839.
[80] K. Zhang, X. Zhou, H. Zhang, and W. Zuo, "Revisiting single image super-resolution under internet environment: Blur kernels and reconstruction algorithms," in *Proc. Pacific Rim Conf. Multimedia*, 2015, pp. 677–687.
[81] M. Irani and S. Peleg, "Motion analysis for image enhancement: Resolution, occlusion, and transparency," *J. Vis. Commun. Image Representation*, vol. 4, no. 4, pp. 324–335, 1993.
[82] N. Zhao, Q. Wei, A. Basarab, N. Dobigeon, D. Kouamé, and J.-Y. Tourneret, "Fast single image super-resolution using a new analytical solution for ℓ_2-ℓ_2 problems," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3683–3697, May 2016.
[83] K. Zhang, L. Van Gool , and R. Timofte, "Deep unfolding network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3217–3226.
[84] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 286–301.
[85] Z. Li, J. Yang, Z. Liu, X. Yang, G. Jeon, and W. Wu, "Feedback network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3867–3876.
[86] J. W. Soh, S. Cho, and N. I. Cho, "Meta-transfer learning for zero-shot super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3516–3525.
[87] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang, "Second-order attention network for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11065–11074.
[88] H. S. Malvar, L.-W. He, and R. Cutler, "High-quality linear interpolation for demosaicing of bayer-patterned color images," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 3, 2004, pp. 485–488.
[89] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand, "Deep joint demosaicking and denoising," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–12, 2016.
[90] L. Liu, X. Jia, J. Liu, and Q. Tian, "Joint demosaicing and denoising with self guidance," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2240–2249.
[91] J. Wu, R. Timofte, and L. Van Gool , "Demosaicing based on directional difference regression and efficient regression priors," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3862–3874, Aug. 2016.
[92] F. Kokkinos and S. Lefkimmiatis, "Iterative joint image demosaicking and denoising using a residual denoising network," *IEEE Trans. Image Process.*, vol. 28, no. 8, pp. 4177–4188, Aug. 2019.
[93] Y. Guo, Q. Jin, G. Facciolo, T. Zeng, and J.-M. Morel, "Residual learning for effective joint demosaicing-denoising," 2020, *arXiv: 2009.06205*.
[94] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 2272–2279.
[95] W. Ye and K.-K. Ma, "Color image demosaicing using iterative residual interpolation," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5879–5891, Dec. 2015.
[96] D. Kiku, Y. Monno, M. Tanaka, and M. Okutomi, "Beyond color difference: Residual interpolation for color image demosaicking," *IEEE Trans. Image Process.*, vol. 25, no. 3, pp. 1288–1300, Mar. 2016.
[97] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–11.

Kai Zhang received the PhD degree from the School of Computer Science and Technology, Harbin Institute of Technology, China, in 2019. From July 2015 to July 2017, he was a research assistant, and from July 2018 to April 2019 he was with the Department of Computing, The Hong Kong Polytechnic University. He is currently a postdoctoral researcher at Computer Vision Laboratory, ETH Zurich, Switzerland, with Prof. Luc Van Gool and Dr. Radu Timofte. His research interests include machine learning and image processing.



Yawei Li received the BE and BEcon degrees in 2014 and the ME degree in 2017 both from the University of Electronic Science and Technology of China. He is currently working toward the PhD degree at Computer Vision Laboratory, ETH Zurich, supervised by Prof. Luc Van Gool and Dr. Radu Timofte. His research interests include image restoration and enhancement, model acceleration, and network compression.



Wangmeng Zuo (Senior Member, IEEE) received the PhD degree in computer application technology from the Harbin Institute of Technology, China, in 2007. From 2004 to 2006, he was a research assistant with the Department of Computing, The Hong Kong Polytechnic University. From 2009 to 2010, he was a visiting professor with Microsoft Research Asia. He is currently a professor at the School of Computer Science and Technology, Harbin Institute of Technology. He has authored or coauthored more than 80 papers in top-tier academic journals and conferences. His current research interests include image enhancement and restoration, image generation and editing, visual tracking, object detection, and image classification. He was a tutorial organizer in ECCV 2016, an associate editor for the *IET Biometrics*, and the guest editor of the *Neurocomputing*, *Pattern Recognition*, *IEEE Transactions on Circuits and Systems for Video Technology*, and *IEEE Transactions on Neural Networks and Learning Systems*.

Lei Zhang (Fellow, IEEE) received the BSc degree from the Shenyang Institute of Aeronautical Engineering, Shenyang, China, in 1995, and the MSc and PhD degrees in control theory and engineering from Northwestern Polytechnical University, Xian, China, in 1998 and 2001, respectively. From 2001 to 2002, he was a research associate with the Department of Computing, The Hong Kong Polytechnic University. From January 2003 to January 2006, he was a postdoctoral fellow with the Department of Electrical and Computer Engineering, McMaster University, Canada. In 2006, he joined the Department of Computing, The Hong Kong Polytechnic University, as an assistant professor. Since July 2017, he has been the chair professor with the Department of Computing, The Hong Kong Polytechnic University. He has authored or coauthored more than 200 papers in his research interests, which include computer vision, image and video analysis, pattern recognition, and biometrics. As of 2020, his publications have been cited more than 54 000 times in literature. He is a currently a senior associate editor for the *IEEE Trans. on Image Processing*, and is currently or was an associate editor for the *IEEE Trans. on Pattern Analysis and Machine Intelligence*, *SIAM Journal of Imaging Sciences*, *IEEE Trans. on CSVT*, and *Image and Vision Computing*. From 2015 to 2019, he was a Clarivate Analytics Highly Cited Researcher.

Luc Van Gool received the degree in electromechanical engineering from the Katholieke Universiteit Leuven, Leuven, Belgium, in 1981. He is currently a professor at the Katholieke Universiteit Leuven, Belgium, and the ETH, Zurich, Switzerland. He leads computer vision research with both places, where he also teaches computer vision. He has authored more than 200 papers in this field. His research interests mainly include 3D reconstruction and modeling, object recognition, tracking, and gesture analysis. He was a program committee member of several major computer vision conferences. He was the recipient of several best paper awards. He is a cofounder of five spin-off companies.

Radu Timofte (Member, IEEE) received the PhD degree in electrical engineering from the KU Leuven, Belgium in 2013. From 2013 to 2016, he was a postdoc at Computer Vision Laboratory, ETH Zurich, Switzerland. He is currently a group leader and a lecturer with Computer Vision Lab. His current research interests include deep learning, implicit models, compression, tracking, restoration, and enhancement. He is currently an editorial board member of top journals, including the the *IEEE Transaction on Pattern Analysis and Machine Intelligence*, *Elsevier Neurocomputing*, *Elsevier Computer Vision and Image Understanding*, *SIAM Journal on Imaging Sciences,* and was the area chair for top conferences, including the CVPR 2021, IJCAI 2021, ECCV 2020, ACCV 2020, and ICCV 2019. His was the recipient of several awards. He is a co-founder of Merantix and a co-organizer of NTIRE, CLIC, AIM, and PIRM events.