# Plug-and-Play Image Restoration with Deep Denoiser Prior — **Source link** 🔗

Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang ...+2 more authors

**Institutions:** ETH Zurich, Harbin Institute of Technology, Hong Kong Polytechnic University

Related papers:

- A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics

- Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising

- Learning Deep CNN Denoiser Prior for Image Restoration

- Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering

- NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study

# 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 $\sigma$. 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.

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,

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

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

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

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 or prior term $\lambda\mathcal{R}(\mathbf{x})$ with regularization parameter $\lambda$. Specifically, the data term guarantees the solution accords with the degradation process, while the prior term alleviates the ill-posedness of the problem 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

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

where $\Theta$ 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).

K. Zhang, Y. Li, L. Van Gool and R. Timofte are with the Computer Vision Lab, ETH Zürich, Zürich 8092, Switzerland (e-mail: kai.zhang@vision.ee.ethz.ch; yawei.li@vision.ee.ethz.ch; vangool@vision.ee.ethz.ch; timofter@vision.ee.ethz.ch).
W. Zuo is with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China (e-mail: cswmzuo@gmail.com).
L. Zhang is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China (e-mail: cslzhang@comp.polyu.edu.hk).

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 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 IR-CNN [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 the denoising subproblem for 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 in-painting. 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 non-linear 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 in-painting and deblurring. In [35], Gavaskar and Chaudhury investigated the convergence of ISTA-based plug-and-play IR with non-local means denoiser.
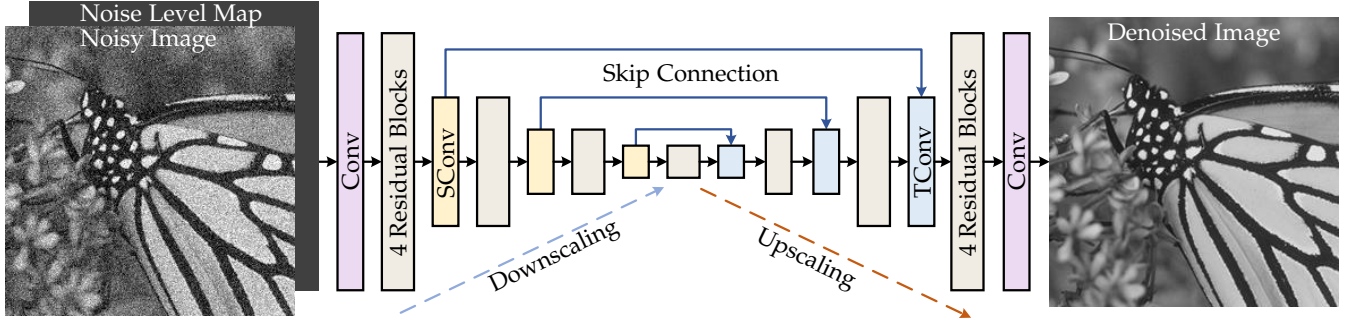
Fig. 1. The architecture of the proposed DRUNet denoiser prior. DRUNet takes an additional noise level map as input and combines U-Net [20] and ResNet [36]. "SConv" and "TConv" represent strided convolution and transposed convolution, respectively.

## 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 denoiser 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 Sec. 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 the following 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 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 does not focus on designing new denoising network architecture. The similar idea of combining U-Net and ResNet can also be found in other works such as [61], [62].

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

| Datasets | Noise Level | BM3D | WNNM | DnCNN | $N^3$Net | NLRN | RNAN | FOCNet | IRCNN | FFDNet | DRUNet |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 15 | 32.37 | 32.70 | 32.86 | – | 33.16 | – | 33.07 | 32.77 | 32.75 | 33.25 |
| Set12 | 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 |
| | 15 | 31.08 | 31.37 | 31.73 | – | 31.88 | – | 31.83 | 31.63 | 31.63 | 31.91 |
| BSD68 | 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 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 $2 \times 2$ strided convolution (SConv) downscaling and $2 \times 2$ 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 $180 \times 180$ [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 $\sigma$. Correspondingly, the noise level map is a uniform map filled with $\sigma$ and has the same spatial size as noisy image. To handle a wide range of noise levels, the noise level $\sigma$ is randomly chosen from $[0, 50]$ during training. 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 $128 \times 128$ 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], $N^3$Net [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 $N^3$Net, 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 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
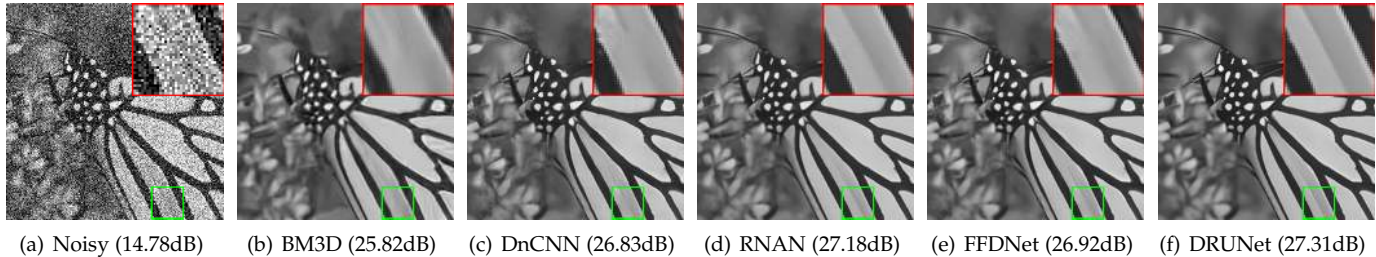
(a) Noisy (14.78dB)  (b) BM3D (25.82dB)  (c) DnCNN (26.83dB)  (d) RNAN (27.18dB)  (e) FFDNet (26.92dB)  (f) DRUNet (27.31dB)

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



(a) Noisy (14.99dB)  (b) BM3D (28.36dB)  (c) DnCNN (28.68dB)  (d) FFDNet (28.75dB)  (e) IRCNN (28.69dB)  (f) DRUNet (29.28dB)
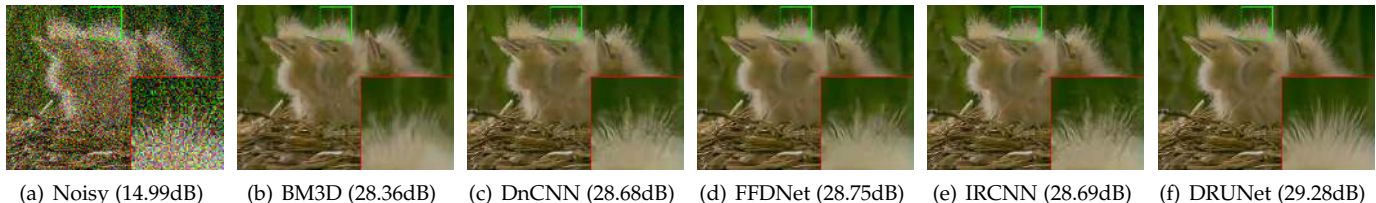
Fig. 3. Color image denoising results of different methods on image "*163085*" from CBSD68 dataset with noise level 50.

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.



(a) Noisy (7.50dB)  (b) FFDNet (20.97dB)  (c) DRUNet (23.55dB)

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

### 3.3.4 Runtime and Maximum GPU Memory Consumption

Table 3 reports the runtime and maximum GPU memory consumption comparison with two representative methods (i.e., DnCNN and RNAN) on images of size $256 \times 256$ and $512 \times 512$ with noise level 50. Note that, for the sake of reducing 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 $100 \times 100$. We also compare RNAN* which sets maximum spatial size to 70,000. As a simple example, RNAN and RNAN* splits an image of size $512 \times 512$ 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 $43 \times 43$. However, it uses a small stride of 7 which would largely increase the computational burden.

From Table 3, one can see that DnCNN achieves the best performance on runtime and memory. While DRUNet has a much better PSNR than DnCNN, it only doubles the runtime and quadruples the maximum GPU memory consumption. In contrast, RNAN is about 60 times slower

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

| 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  |

### 3.3.3 Generalizability to Unseen Noise Level

Fig. 4 provides an example to demonstrate the advantage of bias-free DRUNet over FFDNet. It can be seen that, even trained on noise level range of $[0, 50]$, DRUNet can still perform well on an extremely large unseen noise level of 200. In contrast, FFDNet which was trained on a wider noise level range (i.e., $[0, 75]$) introduces some visual artifacts while having a much lower PSNR than DRUNet. As a result, bias-free DRUNet has a better generalizability to unseen noise level than FFDNet.
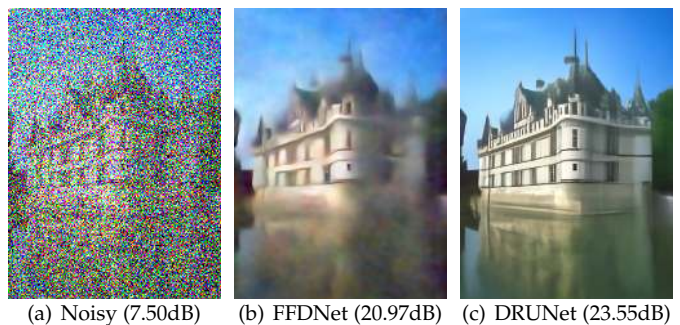
than DnCNN and would dramatically aggravate the maximum GPU memory consumption with the increase of the predefined maximum spatial size. Note that RNAN does not outperform DRUNet in terms of PSNR. Such a phenomenon highlights that the non-local module in RNAN may not be a proper way to improve PSNR and further study is required to improve the runtime and maximum GPU memory consumption.

TABLE 3
Runtime (in seconds) and max GPU memory (in GB) of different methods on images of size 256×256 and 512×512 with noise level 50. 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.

| Metric | Image Size | DnCNN | RNAN | RNAN* | DRUNet |
|---|---|---|---|---|---|
| Runtime | 256×256 | 0.0087 | 0.4675 | 0.4662 | 0.0221 |
| | 512×512 | 0.0314 | 2.1530 | 1.8769 | 0.0733 |
| Memory | 256×256 | 0.0339 | 0.3525 | 2.9373 | 0.2143 |
| | 512×512 | 0.1284 | 0.4240 | 3.2826 | 0.4911 |

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

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

(4) is then solved by minimizing the following problem

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

where $\mu$ 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,

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

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

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

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 $\mu$, regularization parameter $\lambda$ and the total number of iterations $K$.

To guarantee $\mathbf{x}_k$ and $\mathbf{z}_k$ converge to a fixed point, a large $\mu$ is needed, which however requires a large $K$ for convergence. Hence, the common way is to adopt the continuation strategy to gradually increase $\mu$, 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 $\mu$ controls the noise level $\sigma_k(\triangleq \sqrt{\lambda/\mu_k})$ in $k$-th iteration of the denoiser prior. On the other hand, a noise level range of $[0, 50]$ is supposed to be enough for $\sigma_k$. Inspired by such domain knowledge, we can instead set $\sigma_k$ and $\lambda$ to implicitly determine $\mu_k$. Based on the fact that $\mu_k$ should be monotonically increasing, we uniformly sample $\sigma_k$ from a large noise level $\sigma_1$ to a small one $\sigma_K$ in log space. This means that $\mu_k$ can be easily determined via $\mu_k = \lambda/\sigma_k^2$. Following [17], $\sigma_1$ is fixed to 49 while $\sigma_K$ is determined by the image noise level $\sigma$. Since $K$ is user-specified and $\sigma_K$ 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 $\lambda$. Due to the fact that $\lambda$ comes from the prior term and thus should be fixed, we can choose the optimal $\lambda$ by a grid search on a validation dataset. Empirically, $\lambda$ can yield favorable performance from the range of $[0.19, 0.55]$. In this paper,
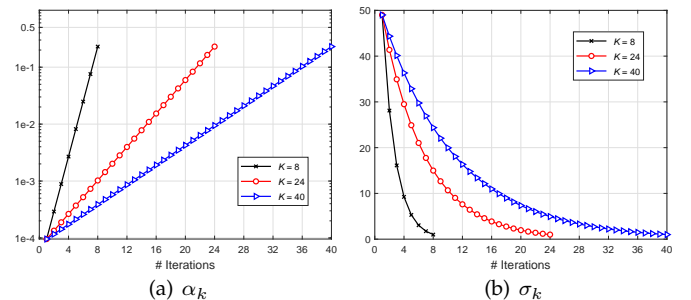


Fig. 5. The values of $\alpha_k$ and $\sigma_k$ at $k$-th iteration with respect to different number of iterations $K$ = 8, 24, and 40.

we fix it to 0.23 unless otherwise specified. It should be noted that since $\lambda$ can be absorbed into $\sigma$ and plays the role of controlling the trade-off between data term and prior term, one can implicitly tune $\lambda$ by multiplying $\sigma$ 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 $\alpha_k$ and $\sigma_k$ 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 [73]. 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~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 $\sigma$, $\sigma_k$ of denoiser prior model at $k$-th iteration for a total of $K$ iterations, trade-off parameter $\lambda$.

**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** $k = 1, 2, \cdots, K$ **do**
3    $\mathbf{x}_k = \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathcal{T}(\mathbf{x})\|^2 + \alpha_k \|\mathbf{x} - \mathbf{z}_{k-1}\|^2$ ; // *Solving data subproblem*
4    $\mathbf{z}_k = Denoiser(\mathbf{x}_k, \sigma_k)$ ; // *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 $\sigma_K$, 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 $\mathcal{F}(\cdot)$. 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.
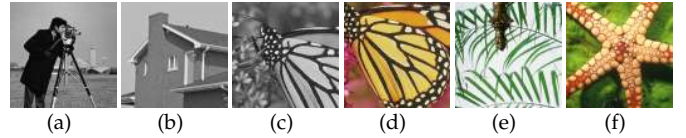


Fig. 6. Six classical testing images. (a) *Cameraman*; (b) *House*; (c) *Monarch*; (d) *Butterfly*; (e) *Leaves*; (f) *Starfish*.

TABLE 4
PSNR(dB) results of different methods on Set6 for image deblurring. The best and second best results are highlighted in red and blue colors, respectively.

| Methods | $\sigma$ | C.man | House | Monarch | Butterfly | Leaves | Starfish |
|---|---|---|---|---|---|---|---|
| | | The second kernel of size 17×17 from [74] | | | | | |
| EPLL | | 29.18 | 32.33 | 27.32 | 24.96 | 23.48 | 28.05 |
| FDN | | 29.09 | 29.75 | 29.13 | 28.06 | 27.04 | 28.12 |
| IRCNN | 2.55 | 31.69 | 35.04 | 32.71 | 33.13 | 33.51 | 33.15 |
| IRCNN+ | | 31.23 | 34.01 | 31.85 | 32.55 | 32.66 | 32.34 |
| DPIR | | 32.05 | 35.82 | 33.38 | 34.26 | 35.19 | 34.21 |
| EPLL | | 24.82 | 28.50 | 23.03 | 20.82 | 20.06 | 24.23 |
| FDN | | 26.18 | 28.01 | 25.86 | 24.76 | 23.91 | 25.21 |
| IRCNN | 7.65 | 27.70 | 31.94 | 28.23 | 28.73 | 28.63 | 28.76 |
| IRCNN+ | | 27.64 | 31.00 | 27.66 | 28.52 | 28.17 | 28.50 |
| DPIR | | 28.17 | 32.79 | 28.48 | 29.52 | 30.11 | 29.83 |
| | | The fourth kernel of size 27×27 from [74] | | | | | |
| EPLL | | 27.85 | 28.13 | 22.92 | 20.55 | 19.22 | 24.84 |
| FDN | | 28.78 | 29.29 | 28.60 | 27.40 | 26.51 | 27.48 |
| IRCNN | 2.55 | 31.56 | 34.73 | 32.42 | 32.74 | 33.22 | 32.53 |
| IRCNN+ | | 31.29 | 34.17 | 31.82 | 32.48 | 33.59 | 32.18 |
| DPIR | | 31.97 | 35.52 | 32.99 | 34.18 | 35.12 | 33.91 |
| EPLL | | 24.31 | 26.02 | 20.86 | 18.64 | 17.54 | 22.47 |
| FDN | | 26.13 | 27.41 | 25.39 | 24.27 | 23.53 | 24.71 |
| IRCNN | 7.65 | 27.58 | 31.55 | 27.99 | 28.53 | 28.45 | 28.42 |
| IRCNN+ | | 27.49 | 30.80 | 27.54 | 28.40 | 28.14 | 28.20 |
| DPIR | | 27.99 | 32.87 | 28.27 | 29.45 | 30.27 | 29.46 |

(a) Blurry image    (b) EPLL (17.54dB)    (c) FDN (23.53dB)    (d) IRCNN (28.45dB)    (e) IRCNN+ (28.14dB)    (f) DPIR (30.27dB)
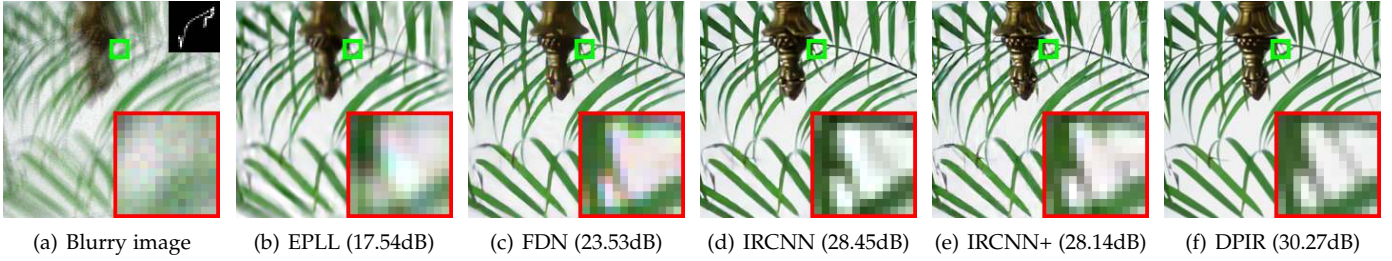
Fig. 7. Visual results comparison of different deblurring methods on *Leaves*. The blur kernel is visualized in the upper right corner of the blurry image. The noise level is 7.65(3%).
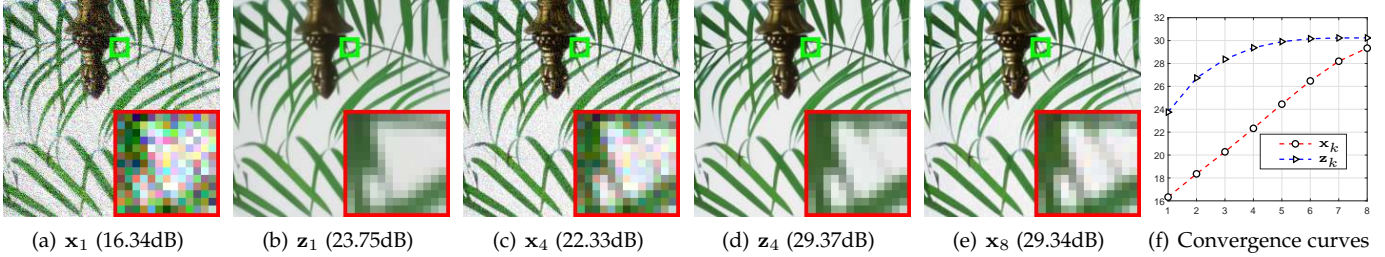


(a) $\mathbf{x}_1$ (16.34dB)    (b) $\mathbf{z}_1$ (23.75dB)    (c) $\mathbf{x}_4$ (22.33dB)    (d) $\mathbf{z}_4$ (29.37dB)    (e) $\mathbf{x}_8$ (29.34dB)    (f) Convergence curves

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).

### 5.1.1 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 [74]. 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 4, 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 $\sigma$. For the parameters $K$ and $\sigma_K$, they are set to 8 and $\sigma$, respectively. For $\mathbf{z}_0$, it is initialized as $\mathbf{y}$.

To evaluate the effectiveness of the proposed DPIR, we choose three representative methods for comparison, including model-based method EPLL [4], learning-based method FDN [75], and plug-and-play method IRCNN and IRCNN+. Table 4 summarizes the PSNR results on Set6. As one can see, 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 EPLL and FDN tend to smooth out fine details and generate color artifacts. Although IRCNN and

TABLE 5
PSNR results with different combinations of $K$ and $\sigma_1$ 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 |

IRCNN+ avoid the color artifacts, it fails to recover the fine details. In contrast, the proposed DPIR can recover image sharpness and naturalness.

### 5.1.2 Intermediate Results and Convergence

Figs. 8(a)-(e) provide the visual results of $\mathbf{x}_k$ and $\mathbf{z}_k$ at different iterations on the testing image from Fig. 7, while Fig. 8(f) 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. 8(f), $\mathbf{x}_k$ and $\mathbf{z}_k$ enjoy a fast convergence to the fixed point.

### 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 $\sigma_1$ to be 49, it is interesting to investigate the performance with other settings. Table 5 reports the PSNR results with different combinations of $K$ and $\sigma_1$ on the testing image from Fig. 7. One can see that larger $\sigma_1$, such as 39 and 49, could result in better PSNR results. On the other hand, if $\sigma_1$ 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 $\sigma_1$ play an important role for the trade-off between efficiency and effectiveness.

## 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 [76], [77]. 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], [78] 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 $s$, $\gamma$ 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 [79], by assuming the convolution is carried out with circular boundary conditions as in deblurring, the closed-form solution is given by

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

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 multiplication, 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 [80]. It is easy to verify that (15) 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 [80]. 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., $\alpha_k$) 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 $\sigma_K$ are set to 24 and $\max(\sigma, s)$, respectively. For the parameters in (12) and (13), $\gamma$ 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 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. 9. Following [80], 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.



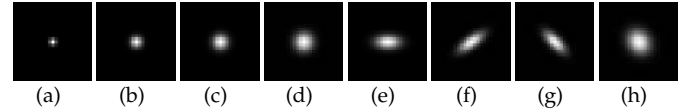| (a) | (b) | (c) | (d) | (e) | (f) | (g) | (h) |

Fig. 9. The eight testing Gaussian kernels for SISR. (a)-(d) are isotropic Gaussian kernels; (e)-(f) are anisotropic Gaussian kernels.

For the compared methods, we consider the bicubic interpolation method, RCAN [81], MZSR [82], IRCNN and IRCNN+. Specifically, RCAN is the state-of-the-art bicubic degradation based deep model consisting of about 400 layers. Note that we do not retrain the RCAN model to handle the testing degradation cases as it lacks 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 6 reports the average PSNR(dB) results of different methods for bicubic degradation and classical degradation on color BSD68 dataset. From Table 6, we can have several observations. First, as expected, RCAN achieves the best results on bicubic degradation with $\sigma = 0$ but loses effectiveness when the true degradation deviates from the assumed one. 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.

TABLE 6
Average PSNR(dB) results of different methods for single image super-resolution on CBSD68 dataset. The best and second best results are highlighted in red and blue colors, respectively.

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



(a) Bicubic (24.82dB)  (b) RCAN (24.61dB)  (c) MZSR (27.34dB)  (d) IRCNN (26.89dB)  (e) IRCNN+ (28.65dB)  (f) DPIR (29.12dB)
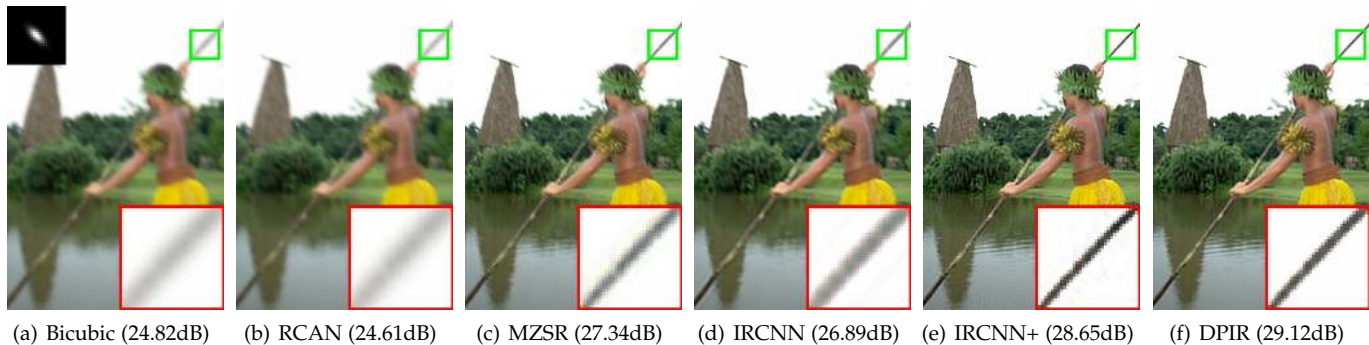
Fig. 10. Visual results comparison of different SISR methods on an image corrupted by classical degradation model. The kernel is shown on the upper-left corner of the bicubicly interpolated LR image. The scale factor is 2.
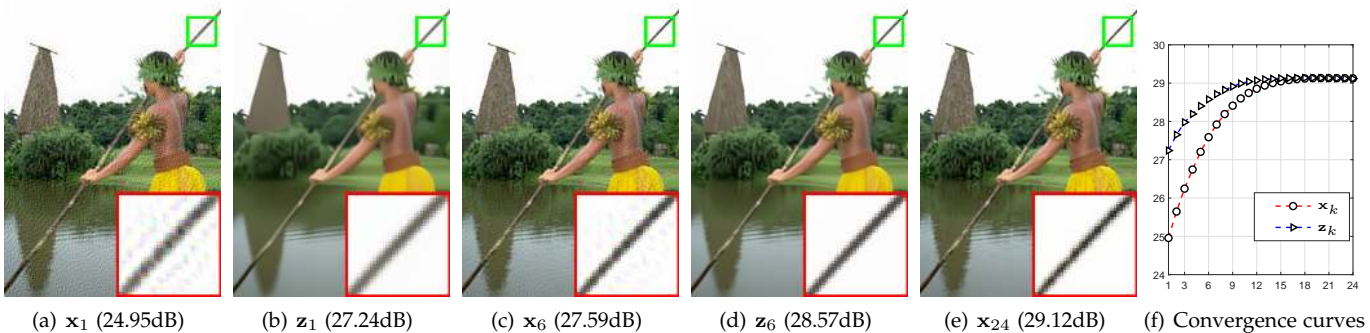


(a) $\mathbf{x}_1$ (24.95dB)  (b) $\mathbf{z}_1$ (27.24dB)  (c) $\mathbf{x}_6$ (27.59dB)  (d) $\mathbf{z}_6$ (28.57dB)  (e) $\mathbf{x}_{24}$ (29.12dB)  (f) Convergence curves

Fig. 11. (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).

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. 10 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.

### 5.2.2 Intermediate Results and Convergence

Fig. 11(a)-(e) 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. 10. 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. 11(f), from which we can see that $\mathbf{x}_k$ and $\mathbf{z}_k$ converge quickly to the fixed point.

TABLE 7
Demosaicing results of different methods on Kodak and McMaster datasets. The best and second best results are highlighted in red and blue colors, respectively.

| Datasets | Matlab | DDR | DeepJoint | MMNet | RNAN | LSSC | IRI | FlexISP | IRCNN | IRCNN+ | DPIR |
|----------|--------|-----|-----------|-------|------|------|-----|---------|-------|--------|------|
| Kodak | 35.78 | 41.11 | 42.0 | 40.19 | 43.16 | 41.43 | 39.23 | 38.52 | 40.29 | 40.80 | 42.68 |
| McMaster | 34.43 | 37.12 | 39.14 | 37.09 | 39.70 | 36.15 | 36.90 | 36.87 | 37.45 | 37.79 | 39.39 |



(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)
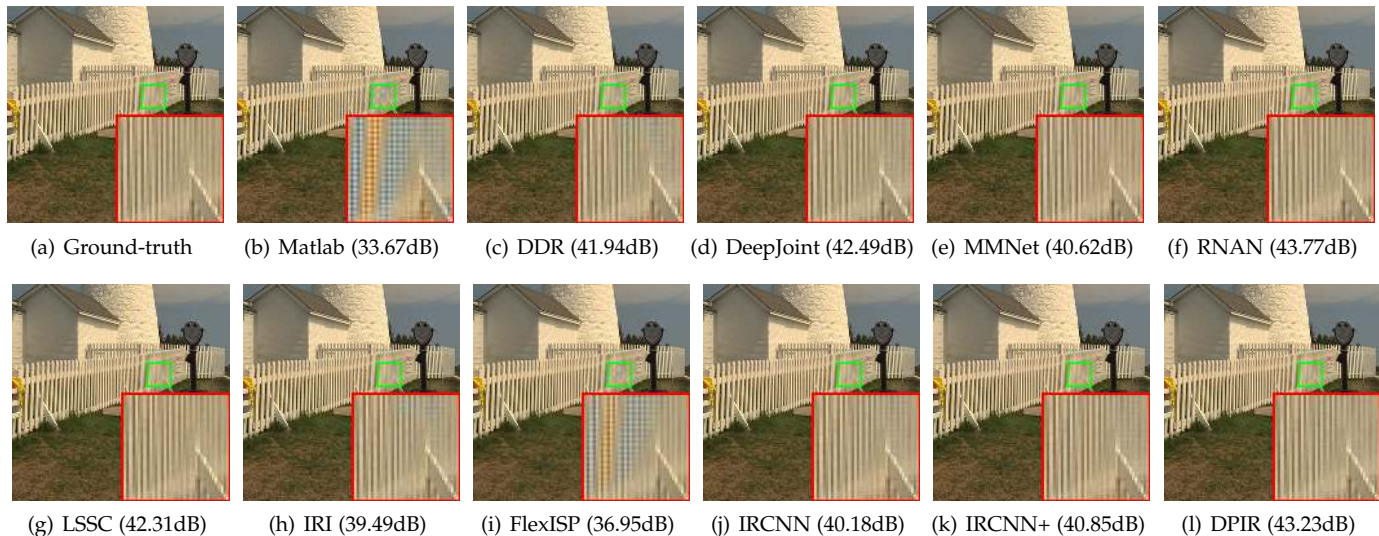
Fig. 12. Visual results comparison of different demosaicing methods on image *kodim19* from Kodak dataset.

### 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. The degradation model of mosaiced image can be expressed as

$$\mathbf{y} = \mathbf{M} \odot \mathbf{x} \qquad (15)$$

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

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

In this paper, we consider the commonly-used Bayer CFA pattern with RGGB arrangement. For the parameters $K$ and $\sigma_K$, 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 [83],

directional difference regression (DDR) [84], deep unfolding majorization-minimization network (MMNet) [85], deep joint demosaicing and denoising (DeepJoint) [86], very deep residual non-local attention network (RNAN) [58] learned simultaneous sparse coding (LSSC) [87], iterative residual interpolation (IRI) [88], minimized-Laplacian residual interpolation (MLRI) [89], primal-dual algorithm with CBM3D denoiser prior (FlexISP) [15], IRCNN and IRCNN+. Note that DDR, MMNet, DeepJoint, and RNAN are learning-based methods, while LSSC, IRI, MLRI, FlexISP, IRCNN, IRCNN+, and DPIR are model-based methods.

Table 7 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. 12 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. 13(a)-(e) 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
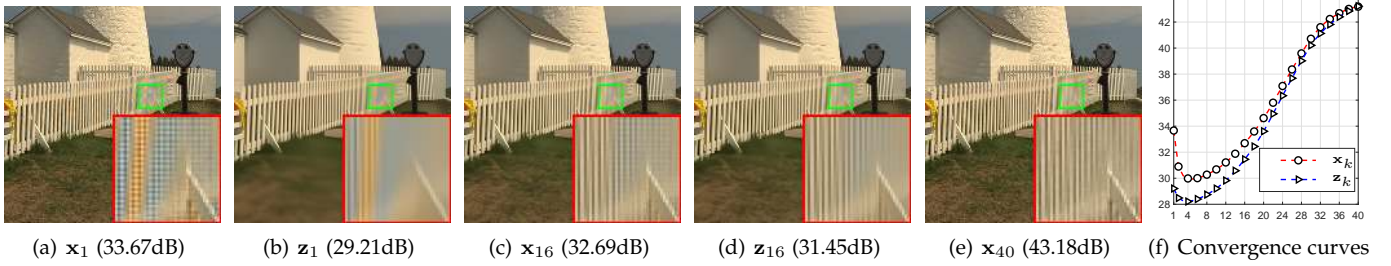
(a) $\mathbf{x}_1$ (33.67dB) (b) $\mathbf{z}_1$ (29.21dB) (c) $\mathbf{x}_{16}$ (32.69dB) (d) $\mathbf{z}_{16}$ (31.45dB) (e) $\mathbf{x}_{40}$ (43.18dB) (f) Convergence curves

Fig. 13. (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).

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. 13(f) 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. 14 shows the noise histogram of $\mathbf{x}_1$ and $\mathbf{x}_8$ in Fig. 8 for deblurring, $\mathbf{x}_1$ and $\mathbf{x}_{24}$ in Fig. 11 for super-resolution, and $\mathbf{x}_1$ and $\mathbf{x}_{40}$ in Fig. 13 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. 14(a) 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.



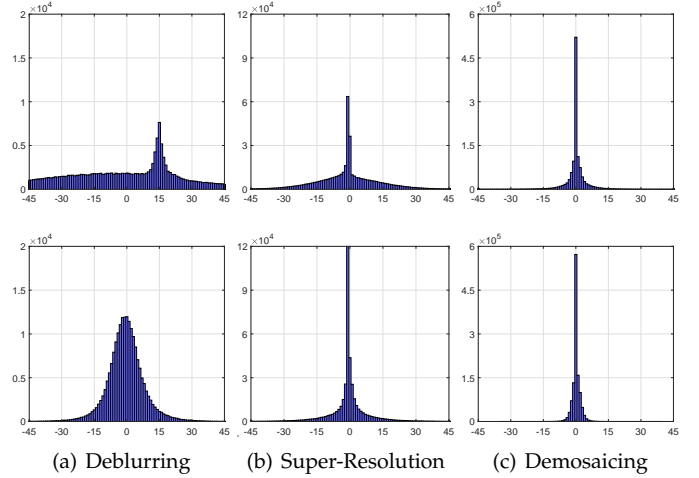(a) Deblurring (b) Super-Resolution (c) Demosaicing

Fig. 14. 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. 11 and $\mathbf{x}_1$ and $\mathbf{x}_{40}$ in Fig. 13.

## 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 [90], for effective image restoration.

# 8 ACKNOWLEDGEMENTS

## REFERENCES

[1] W. H. Richardson, "Bayesian-based iterative method of image restoration," *JOSA*, vol. 62, no. 1, pp. 55–59, 1972.

[2] H. C. Andrews and B. R. Hunt, "Digital image restoration," *Prentice-Hall Signal Processing Series, Englewood Cliffs: Prentice-Hall, 1977*, vol. 1, 1977.

[3] S. Roth and M. J. Black, "Fields of experts," *International Journal of Computer Vision*, 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 *IEEE International Conference on Computer Vision*, 2011, pp. 479–486.

[5] M. F. Tappen, "Utilizing variational optimization to learn markov random fields," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[6] A. Barbu, "Training an active random field for real-time image denoising," *IEEE Transactions on Image Processing*, vol. 18, no. 11, pp. 2451–2462, 2009.

[7] J. Sun and M. F. Tappen, "Separable markov random field model and its applications in low level vision," *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 402–407, 2013.

[8] U. Schmidt and S. Roth, "Shrinkage fields for effective image restoration," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2774–2781.

[9] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1256–1272, 2016.

[10] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *IEEE Conference on Computer Vision and Pattern Recognition*, 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 and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[12] D. Geman and C. Yang, "Nonlinear image recovery with half-quadratic regularization," *IEEE Transactions on Image Processing*, vol. 4, no. 7, pp. 932–946, 1995.

[13] N. Parikh, S. P. Boyd *et al.*, "Proximal algorithms," *Foundations and Trends in optimization*, 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 *Workshop on Information Theoretic Methods in Science and Engineering*, 2010, pp. 16–18.

[15] F. Heide, M. Steinberger, Y.-T. Tsai, M. Rouf, D. Pajak, D. Reddy, O. Gallo, J. Liu, W. Heidrich, K. Egiazarian *et al.*, "FlexISP: A flexible camera image processing framework," *ACM Transactions on Graphics*, vol. 33, no. 6, p. 231, 2014.

[16] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *IEEE Global Conference on Signal and Information Processing*, 2013, pp. 945–948.

[17] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," in *IEEE Conference on Computer Vision and Pattern Recognition*, July 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 Transactions on Image Processing*, vol. 27, no. 9, pp. 4608–4622, 2018.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 234–241.

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

[22] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 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 Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

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

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

[26] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[27] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *Journal of Mathematical Imaging and Vision*, 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 *IEEE International Conference on Image Processing*. IEEE, 2016, pp. 3518–3522.

[29] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "From denoising to compressed sensing," *IEEE Transactions on Information Theory*, vol. 62, no. 9, pp. 5117–5144, 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 Transactions on Image Processing*, vol. 12, no. 11, pp. 1338–1351, 2003.

[31] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-play admm for image restoration: Fixed-point convergence and applications," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 84–98, 2016.

[32] U. S. Kamilov, H. Mansour, and B. Wohlberg, "A plug-and-play priors approach for solving nonlinear imaging inverse problems," *IEEE Signal Processing Letters*, vol. 24, no. 12, pp. 1872–1876, 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 *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 7665–7669.

[34] N. Yair and T. Michaeli, "Multi-scale weighted nuclear norm image restoration," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3165–3174.

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

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

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

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

[39] S. Gu, R. Timofte, and L. Van Gool, "Integrating local and non-local denoiser priors for image restoration," in *International Conference on Pattern Recognition*, 2018, pp. 2923–2928.

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

[41] Z. Li and J. Wu, "Learning deep cnn denoiser priors for depth image inpainting," *Applied Sciences*, vol. 9, no. 6, p. 1103, 2019.

[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 *International Conference on Machine Learning*, 2019, pp. 5546–5557.

[43] Y. Sun, J. Liu, and U. Kamilov, "Block coordinate regularization by denoising," in *Advances in Neural Information Processing Systems*, 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 Transactions on Image Processing*, pp. 3142–3155, 2017.

[45] J. Zhang and B. Ghanem, "ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1828–1837.

[46] H. K. Aggarwal, M. P. Mani, and M. Jacob, "Modl: Model-based deep learning architecture for inverse problems," *IEEE transactions on medical imaging*, vol. 38, no. 2, pp. 394–405, 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 Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2305–2318, 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, p. 034005, 2020.

[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 *IEEE International Conference on Computer Vision*, vol. 2, July 2001, pp. 416–423.

[50] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2noise: Learning image restoration without clean data," in *International Conference on Machine Learning*, 2018, pp. 2965–2974.

[51] A. Krull, T.-O. Buchholz, and F. Jug, "Noise2void-learning denoising from single noisy images," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2129–2137.

[52] J. Batson and L. Royer, "Noise2self: Blind denoising by self-supervision," in *International Conference on Machine Learning*, 2019, pp. 524–533.

[53] S. Guo, Z. Yan, K. Zhang, W. Zuo, and L. Zhang, "Toward convolutional blind denoising of real photographs," in *IEEE Conference on Computer Vision and Pattern Recognition*, 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 *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 036–11 045.

[55] A. Abdelhamed, M. A. Brubaker, and M. S. Brown, "Noise flow: Noise modeling with conditional normalizing flows," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3165–3173.

[56] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao, "Cycleisp: Real image restoration via improved data synthesis," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[57] S. Lefkimmiatis, "Non-local color image denoising with convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 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 *International Conference on Learning Representations*, 2019.

[59] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang, "Non-local recurrent network for image restoration," in *Advances in Neural Information Processing Systems*, 2018, pp. 1673–1682.

[60] T. Plötz and S. Roth, "Neural nearest neighbors networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 1087–1098.

[61] Z. Zhang, Q. Liu, and Y. Wang, "Road extraction by deep residual u-net," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 5, pp. 749–753, 2018.

[62] G. Venkatesh, Y. Naresh, S. Little, and N. E. OConnor, "A deep residual architecture for skin lesion segmentation," in *Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*. Springer, 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 *IEEE Conference on Computer Vision and Pattern Recognition 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 *International Conference on Learning Representations*, 2019.

[65] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang, "Waterloo exploration database: New challenges for image quality assessment models," *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 1004–1016, 2017.

[66] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, vol. 3, July 2017, pp. 126–135.

[67] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference for Learning Representations*, 2015.

[68] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[69] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, "Residual non-local attention networks for image restoration," in *International Conference on Learning Representations*, 2019.

[70] X. Jia, S. Liu, X. Feng, and L. Zhang, "Focnet: A fractional optimal control network for image denoising," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6054–6063.

[71] R. Franzen, "Kodak lossless true color image suite," *source: http://r0k. us/graphics/kodak*, 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," *Journal of Electronic Imaging*, vol. 20, no. 2, p. 023016, 2011.

[73] R. Timofte, R. Rothe, and L. Van Gool, "Seven ways to improve example-based single image super resolution," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1865–1873.

[74] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Understanding and evaluating blind deconvolution algorithms," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1964–1971.

[75] J. Kruse, C. Rother, and U. Schmidt, "Learning to push the limits of efficient fft-based image deconvolution," in *IEEE International Conference on Computer Vision*, 2017, pp. 4586–4594.

[76] N. Efrat, D. Glasner, A. Apartsin, B. Nadler, and A. Levin, "Accurate blur models vs. image priors in single image super-resolution," in *IEEE International Conference on Computer Vision*, 2013, pp. 2832–2839.

[77] K. Zhang, X. Zhou, H. Zhang, and W. Zuo, "Revisiting single image super-resolution under internet environment: blur kernels and reconstruction algorithms," in *Pacific Rim Conference on Multimedia*, 2015, pp. 677–687.

[78] M. Irani and S. Peleg, "Motion analysis for image enhancement: Resolution, occlusion, and transparency," *Journal of Visual Communication and Image Representation*, vol. 4, no. 4, pp. 324–335, 1993.

[79] 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 $\ell_2$-$\ell_2$ problems," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3683–3697, 2016.

[80] K. Zhang, L. Van Gool, and R. Timofte, "Deep unfolding network for image super-resolution," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3217–3226.

[81] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 286–301.

[82] J. W. Soh, S. Cho, and N. I. Cho, "Meta-transfer learning for zero-shot super-resolution," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3516–3525.

[83] H. S. Malvar, L.-w. He, and R. Cutler, "High-quality linear interpolation for demosaicing of bayer-patterned color images," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, 2004, pp. iii–485.

[84] J. Wu, R. Timofte, and L. Van Gool, "Demosaicing based on directional difference regression and efficient regression priors," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3862–3874, 2016.

[85] F. Kokkinos and S. Lefkimmiatis, "Iterative joint image demosaicking and denoising using a residual denoising network," *IEEE Transactions on Image Processing*, vol. 28, no. 8, pp. 4177–4188, 2019.

[86] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand, "Deep joint demosaicking and denoising," *ACM Transactions on Graphics*, vol. 35, no. 6, pp. 1–12, 2016.

[87] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *IEEE International Conference on Computer Vision*, 2009, pp. 2272–2279.

[88] W. Ye and K.-K. Ma, "Color image demosaicing using iterative residual interpolation," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5879–5891, 2015.

[89] D. Kiku, Y. Monno, M. Tanaka, and M. Okutomi, "Beyond color difference: Residual interpolation for color image demosaicking," *IEEE Transactions on Image Processing*, vol. 25, no. 3, pp. 1288–1300, 2016.

[90] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," in *International Conference on Learning Representations*, 2018.