Joint demosaicking and denoising benefits from a two-stage training strategy

Yu Guo, Qiyu Jin, Jean-Michel Morel, Tieyong Zeng, Gabriele Facciolo

Joint Demosaicking and Denoising Benefits from a Two-stage Training Strategy

Yu Guo[a], Qiyu Jin[a,*], Jean-Michel Morel[b], Tieyong Zeng[c], Gabriele Facciolo[b]

[a] *School of Mathematical Science, Inner Mongolia University, Hohhot, China*
[b] *Centre Borelli, CNRS, ENS Paris-Saclay, Université Paris-Saclay, France.*
[c] *Department of Mathematics, The Chinese University of Hong Kong, Satin, Hong Kong*

Abstract

Image demosaicking and denoising are the first two key steps of the color image production pipeline. The classical processing sequence has for a long time consisted of applying denoising first, and then demosaicking. Applying the operations in this order leads to oversmoothing and checkerboard effects. Yet, it was difficult to change this order, because once the image is demosaicked, the statistical properties of the noise are dramatically changed and hard to handle by traditional denoising models. In this paper, we address this problem by a hybrid machine learning method. We invert the traditional color filter array (CFA) processing pipeline by first demosaicking and then denoising. Our demosaicking algorithm, trained on noiseless images, combines a traditional method and a residual convolutional neural network (CNN). This first stage retains all known information, which is the key point to obtain faithful final results. The noisy demosaicked image is then passed through a second CNN restoring a noiseless full-color image. This pipeline order completely avoids checkerboard effects and restores fine image detail. Although CNNs can be trained to solve jointly demosaicking-denoising end-to-end, we find that this two-stage training performs better and is less prone to failure. It is shown experimentally to improve on the state of the art, both quantitatively and in terms of visual quality.

Keywords: Demosaicking, denoising, pipeline, convolutional neural networks, residual.

1. Introduction

The objective of demosaicking is to build a full-color image from four spatially undersampled color channels. Indeed, digital cameras can only capture one color information on each pixel through a single monochrome sensor, and

*Corresponding author.
Email addresses: yuguomath@aliyun.com (Yu Guo), qyjin2015@aliyun.com (Qiyu Jin), morel@ens-paris-saclay.fr (Jean-Michel Morel), zeng@math.cuhk.edu.hk (Tieyong Zeng), gabriele.facciolo@ens-paris-saclay.fr (Gabriele Facciolo)

(a) Bayer pattern (b) CFA image (c) Mosaic image (d) Color image

Figure 1: The image shows the raw data collected by the sensor and the color filter arrays of the Bayer pattern.

most of them use color filter arrays (CFA) such as the Bayer pattern [1] (shown in Figure 1) to obtain images. The raw data collected in this way is missing two-thirds of pixels and is contaminated by noise. Hence, image demosaicking, *i.e.* the task of reconstructing a full-color image from the incomplete raw data is a typical ill-posed problem.

The conventional method for processing noisy raw sensor data has been to perform denoising and demosaicking as two independent steps. Since demosaicking is a complex interpolation process, the raw noise becomes correlated and anisotropic after demosaicking (see [2] for a detailed discussion), thus losing its independent Poisson noise structure. This means that most classic denoising algorithms are not directly applicable. Indeed, most algorithms rely on the AGWN (additive Gaussian white noise) assumption, which is approximately valid after a simple Anscombe transform has been applied to the raw data. Moreover, most standard demosaicking algorithms with good performance are designed based on the critical noise-free condition. This takes for granted the assumption that the image processing pipeline starts with denoising [3, 4, 5].

However, some researchers have observed that demosaicking first and then denoising yields a better visual quality. Condat [6] proposed to demosaick first and then project the noise into the luminance channel of the reconstructed image before denoising according to the grayscale image. This idea was later refined in [7, 8]. Recently Jin *et al.* [2] improved the "demosaicking first" pipeline via a simple modification of the traditional color denoiser, and gave the corresponding theoretical explanation.

Both pipelines have significant shortcomings. A "denoising first" pipeline removes noise directly on the CFA image. Yet, CFA image denoising differs from the usual grayscale or full-color image denoising. Indeed, CFA image denoising implies subsampling the CFA image into a half-size four-channel RGGB image, which is then denoised. This leads not only to a poor preservation of image details due to the reduced resolution, but also to a loss of the correlation between the red (R), green (G) and blue (B) channels. As a result, the restored image is oversmoothed and checkerboard effects [9] are introduced. On the other hand, the "demosaicking first" pipeline also introduces a thorny issue: It requires denoising a demosaicked residual noise whose statistical properties have been

changed by a complex interpolation, which are hard to model accurately. This was almost impossible for traditional denoising algorithms, but current data-driven deep learning based methods offer new paths to solve this problem. In recent years, deep convolutional neural networks have achieved great success in computer vision and image processing. In image classification and recognition [10, 11], denoising [12, 13, 14, 15, 16], demosaicking [17, 18, 19, 20, 21], super-resolution [22, 23] and other high-level and low-level visual tasks, deep learning methods surpass traditional methods. Since deep learning is data driven, it can find the hidden rules from the data without relying on hand-made filters and a priori knowledge. In this paper, we take advantage of this new flexibility to handle a noise with complex statistical properties, like the one introduced by a "demosaicking first" pipeline.

We therefore implement a "demosaicking first and then denoising" approach by a network with a two-stage training strategy. Convolutional neural networks (CNNs) are first combined with traditional algorithms to obtain an effective demosaicking algorithm. Using this demosaicking as a base, we use another CNN to remove the demosaicked residual noise, whose statistical properties have been changed. Our main contributions are:

- A CNN architecture implementing the "demosaick first and then denoise" pipeline, which effectively restores full-color images from noisy CFA images while preserving more detail and avoiding oversmoothing and checkerboard artifacts.
- Ablation studies show that this architecture and the proposed two-stage training strategy perform better than usual end-to-end approaches, enable a more stable training, and yield state-of-the-art results.
- A modified Inception architecture to implement the two stages of our network. This choice fosters cross-channel information fusion for producing a more accurate estimate of the original image and improves the receptive field to reduce artifacts. In that way, we obtain a lighter network than current state-of-the-art approaches [24, 25] without compromising performance.

The rest of the paper is organized as follows. Section 2 presents related work on demosaicking and denoising. The demosaicking and denoising model is introduced in Section 3. Section 4 provides quantitative and qualitative comparisons with state-of-the-art methods. The concluding remarks are given in Section 5.

2. Related Work

2.1. Demosaicking

Demosaicking is a classic problem with a vast literature. All authors agree that the key to attaining a good demosaicking is to restore the image regions with high-frequency content. Smooth regions are instead easy to interpolate from the available samples. The earliest demosaicking algorithms used methods

such as spline interpolation and bilinear interpolation to process each channel. These methods introduce serious zipper effects. In order to eliminate the artifacts at the image edges, Laroche and Prescott [26] introduced a direction adaptive filter by selecting a preferred direction to interpolate the additional color values according to gradient values. Inspired by this idea, Adams and Hamilton proposed a direction adaptive inter-channel correlation filter [27, 28] under the assumption that derivatives of R, G and B are nearly equal. The G channel interpolation is obtained by a discrete directional Taylor formula involving the second order derivative of either the R or the B channel (see [29]). Once the G channel interpolation was complete, the G channel was taken as a guide image to help the R and B channel interpolation. Many advanced algorithms have still extended the idea of a combination of direction adaptive and inter-channel correlation. In order to make better use of the correlation between channels, Zhang and Wu [30] developed an adaptive filtering method using directional linear minimum mean square error estimation (DLMMSE). Both horizontal and vertical direction interpolations fail to restore the color value when the pixels are located near some edge or in textured regions resulting in zipper artifacts at those areas. In order to solve this issue, Pekkucuksen and Altunbasak [31] decomposed the horizontal and vertical directions into four directions of east, west, south, and north on the basis of [30], and then used the color differences in the four directions to estimate the missing G value. Similarly to [31], Kiku *et al.* [32] proposed RI which calculates four directions' interpolations of R, G and B channels via a Guided Filter [33], and improves the tentative estimates by substituting a residual technique for the HA interpolation [27, 28]. The MLRI [34] and MLRI+wei [35] were the improved versions of RI by minimizing the Laplacian energy of the guided filter. Moreover, ARI [36] united the advantages of RI and MLRI by combining both methods in an iterative process with the most appropriate number of iteration steps at each pixel. These last interpolation algorithms have received a detailed mathematical analysis in [29].

In addition to the above local interpolation algorithms, other classic image processing techniques have been attempted to tackle the problem: algorithms based on non-local similarity [37, 38, 39], wavelet-based algorithms [40, 41], frequency domain based algorithm [42, 43], and dictionary learning based algorithms [44, 45].

Accompanying the wide application of deep learning in the field of image processing, demosaicking algorithms based on deep learning achieved great success and redefined the state-of-the-art. Tan *et al.* [18] addressed the demosaicking problem by learning a deep residual CNN. A two-phase network architecture was designed to reconstruct the G channel first and then estimate the R and B channel using the reconstructed G channel as guide. After calculating the inter-channel correlation coefficients, Cui *et al.* [46] found that R/G and G/B were more relevant than R/B and established a 3-stage CNN structure for demosaicking according to this observation. Instead of using two-phase or three-phase network architecture, Tan *et al.* [19] learned directly the residual between ground truth image and an initial full color image obtained by a fast demosaicking method [47]. This idea combined the traditional method and CNNs to simplify

the network structure for the demosaicking problem. Syu *et al.* [48] used a convolutional neural network to design a demosaicking algorithm, and compared the effects of convolution kernels of different sizes on the reconstruction. At the same time they also designed a new CFA pattern using a data-driven approach. Different from conventional demosaicking CNN methods, Yamaguchi and Ikehara [49] took chrominance images as the output of CNN to improve the result. Higher-Resolution Network (HERN) was proposed by Mei *et al.* [50] to solve the demosaicking problem by learning global information from high resolution data with a feasible GPU memory usage.

2.2. Joint demosaicking and denoising

Since the raw CFA data is altered by noise while most demosaicking algorithms assume a noise-free image, the image processing pipeline often requires a denoising step. Denoising and demosaicking are both ill-posed problems in the pipeline of reconstruction of full color images. In order to reduce artifacts caused by error accumulation, some works have proposed to jointly perform demosaicking and denoising. Condat and Mosaddegh [51] proposed an algorithm based on total variation minimization. Klatzer *et al.* [52] formulated joint demosaicking and denoising problem as an energy minimization problem. Khashabi *et al.* [53] introduced a machine learning method by learning a statistical model from natural images to avoid artifacts. Menon and Calvagno [54] evaluated the noise properties after the space-varying demosaicking method [55] and then proposed a joint demosaicking and denoising one. Tan *et al.* [56] addressed the joint demosaicking and denoising problem as a TV regularization model with multiple effective priors and solved by the alternating direction method of multipliers (ADMM).

The advent of deep learning techniques and the increasing availability of large training data sets, have led to a new generation of state-of-the-art algorithms that are able to reconstruct the full color images from noisy CFA images. Gharbi *et al.* [17] built a huge image database where images were mined from the web and trained a network on it for avoiding zippering or moiré artifacts. Inspired by image regularization methods [57], Kokkinos *et al.* [24, 58] established a novel deep learning architecture that combines a majorization-minimization algorithm with residual denoising networks. Huang *et al.* [59] proposed a lightweight end-to-end network using deep residual learning and aggregated residual transformations. In order to use real data directly, Ehret *et al.* [60] introduced a mosaic-to-mosaic training strategy which doesn't require ground truth RGB data. The proposed framework can be used to fine-tune a pretrained network to a RAW burst. The self-guidance network (SGNet) [20] was proposed according to the fact that the G channel of CFA image contains more information. The G channel is recovered first and works as a guide image to interpolate the R and B channels. In [21], G channel prior features are utilized as guidance to extract and upsample the features of the whole image. Xing and Egiazarian [25] proposed an end-to-end solution for the joint demosaicking, denoising and super-resolution. They showed that merely training the network with mean absolute error loss function yielded superior results.

Satisfactory results have been obtained for joint demosaicking and denoising based on deep learning, but these algorithms all rely on the fitting power of CNNs to solve multiple tasks simultaneously end-to-end. Undoubtedly, this ignores the inter-task correlation, especially the long debated issue of demosaicking and denoising pipeline order.

3. Residual learning for demosaicking and denoising

The biggest obstacle to applying a demosaicking first and then denoising pipeline is the correlated noise resulting from the demosaicking. This is very difficult for model-based denoisers. Using CNNs can attain satisfactory end-to-end results, however these methods neglect the dependency between the demosaicking and denoising tasks. Inspired by [6, 2], we propose a two-stages CNN for reconstructing full-color images from noisy CFA images. In the first stage, we design a demosaicking algorithm that combines traditional methods and deep learning by ignoring the noise. All known information is retained in this stage, which is key to obtain good final results. After the first stage, a noisy full-color image is obtained whose noise statistical properties have been changed by the demosaicking. The second CNN is used to learn to remove the demosaicked residual noise and to effectively recover the underlying textures.

The noisy CFA model is written as

$$Y = M. * (X + \varepsilon), \tag{1}$$

where X is an original full-color image, Y is the noisy CFA (or mosaicked) image, ε is Gaussian noise with zero mean and standard deviation σ, the operator $.*$ denotes the array element-wise multiplication and M denotes the CFA mask. The CFA mask M and its inverse mask are defined as

$$M = \begin{bmatrix} M_R \\ M_G \\ M_B \end{bmatrix} \quad \text{and} \quad IM = \begin{bmatrix} \mathbf{1} - M_R \\ \mathbf{1} - M_G \\ \mathbf{1} - M_B \end{bmatrix}, \tag{2}$$

$$M_R(i,j) = \begin{cases} 0, & \text{if } (i,j) \notin \Omega_R; \\ 1, & \text{if } (i,j) \in \Omega_R, \end{cases}$$

$$M_G(i,j) = \begin{cases} 0, & \text{if } (i,j) \notin \Omega_G; \\ 1, & \text{if } (i,j) \in \Omega_G, \end{cases}$$

$$M_B(i,j) = \begin{cases} 0, & \text{if } (i,j) \notin \Omega_B; \\ 1, & \text{if } (i,j) \in \Omega_B, \end{cases}$$

where $\mathbf{1}(i,j) = 1$, Ω denotes the set of CFA image pixels, Ω_R, Ω_G, $\Omega_B \subseteq \Omega$ are disjoint sets of pixels, which respectively record R, G and B values in the CFA image, and satisfy $\Omega_R \cup \Omega_G \cup \Omega_B = \Omega$.

The first stage considers only the noise-free CFA model

$$Y = M. * X, \tag{3}$$

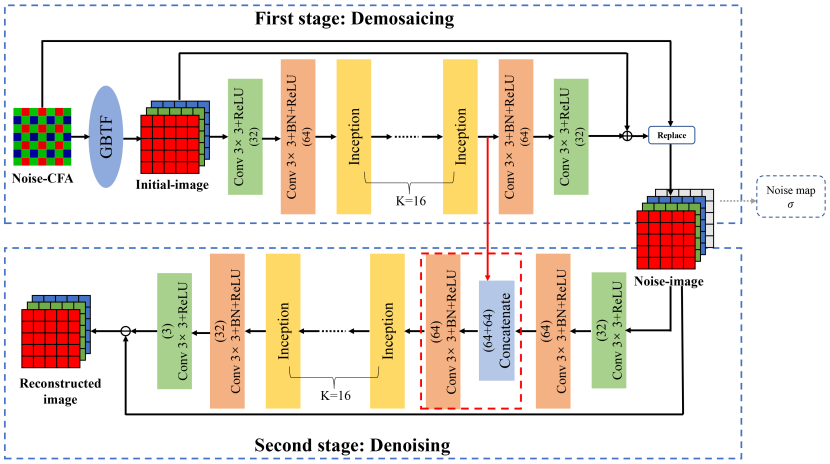


Figure 2: Our two stages CNN architecture for demosaicking-denoising. The first stage takes GBTF to preprocess the CFA image and uses a CNN to learn residuals improving the demosaicking performance of GBTF. In the second stage, when the noisy CFA image is demosaicked, another CNN is used to learn the residual noise in order to reconstruct the finally full-color image. The term "replace" corresponds to Eq. (4).

where X is a full-color image, Y is the noise-free CFA (or mosaicked) image. We first use the GBTF algorithm [31] to obtain a raw demosaicked image $\widehat{X}_{GBTF} = \text{GBTF}(Y)$ and a residual $R_{GBTF} = X - \widehat{X}_{GBTF}$. The residual is then corrected with a CNN. For that we use a modified Inception architecture in order to achieve better performance in learning the residual and get an estimator $\widehat{R}_{GBTF}$ (see Figure 2).

The final full-color image is obtained as

$$\widehat{X}_{DM} = IM. * (\widehat{X}_{GBTF} + \widehat{R}_{GBTF}) + M. * X. \tag{4}$$

The first term in the above equation is the demosaicked image estimated by the CNN and evaluated on the inverse CFA mask IM, while the second term is unaltered input CFA samples on the mask M. The resulting CNN is adapted to demosaick noise-free images. So, applying it to a noisy CFA image, produces a noisy demosaicked image.

To handle noisy CFA images, another stage is needed to remove the noise. Given the trained demosaicking as a basic component, we apply it to model (1) and obtain a noisy full-color image $\widehat{X}_{DM}$ which can be decomposed as

$$\widehat{X}_{DM} = X + \varepsilon_{DM}. \tag{5}$$

Here, ε_{DM} is the residual noise (including artifacts) of the demosaicked image, which is no longer independent identically distributed (I.I.D.), and has complex unknown statistical properties. This would be extremely challenging for traditional denoising models that strongly rely on statistical assumptions, therefore

we use another CNN to learn to extract the residual noise ε_{DM} and obtain the estimator $\widehat{\varepsilon}_{DM}$ (see Figure 2). The final full-color image is reconstructed as

$$\widehat{X}_{DMDN} = \widehat{X}_{DM} - \widehat{\varepsilon}_{DM}. \tag{6}$$

There are several advantages in training separate demosaicking and denoising networks:

- First, the noise-free demosaicking focuses on reconstructing the structure and details in the image without concessions. In addition, the demosaicking network needs not be adapted to each noise level, and all known information is preserved.
- Second, the demosaicked result facilitates the task of the denoiser which has to adapt only to the noise and demosaicking artifacts. As we will see later, training a joint denoising and demosaicking network with equivalent capacity as the demosaicking and denoising networks indeed yields lower quality results.
- Third, the proposed two stage architecture and trainig strategy is more stable at training time than an end-to-end network with equal capacity.

3.1. Demosaicking in a noise-free setting

The CFA images are different from ordinary images as the values of adjacent pixels represent the intensity of different colors. Many of the existing deep learning algorithms subsample the CFA images to four-channel RGGB images and send them to the network. However, this operation reduces the image resolution. Therefore, the network needs to perform functions similar to super-resolution, and cannot only focus on image demosaicking. Some algorithms use bilinear interpolation as preprocessing in order to preserve the spatial arrangement of the samples. However, the bilinear interpolation results are suboptimal and this affects the performance of the convolutional network. In this work, we use the gradient based threshold free (GBTF) method [31], which has superior performance compared to the bilinear interpolation. Improving the network input also alleviates the task for the network. In subsequent ablation experiments, GBTF is shown to better preserve image textures.

After the CFA image is preprocessed, we use a convolutional neural network for residual learning. The network architecture is shown in Figure 2. Syu *et al.* pointed out in their work [48] that convolution kernels of different sizes will affect the reconstruction accuracy. The larger the size of the convolution kernel, the higher the reconstruction accuracy. However, the number of parameters using a 5×5 convolution kernel is 2.7 times that of using a 3×3 convolution kernel. We want to increase the receptive field, but without giving up the lightweight 3×3 convolution kernels. In the image demosaicking task, due to the lack of color information, the full-color image reconstruction must make full use of the correlation of the three RGB channels. Therefore, the degree of cross-channel information fusion determines the performance of the demosaicking algorithm. In

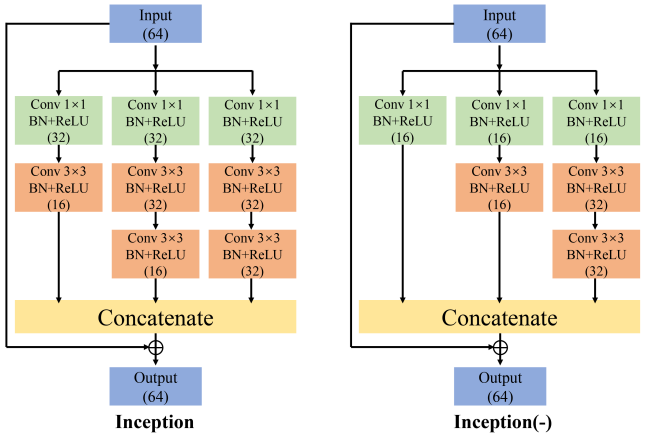


Figure 3: Architecture of the Inception block. In order to get a better cross-channel fusion and a larger receptive field, we use 1×1 convolution kernels and three-way branches to reduce the parameters while strengthening the fusion of cross-channel information. This is extremely important for demosaicking.

order to get a better cross-channel fusion and a larger receptive field, we propose modifying the architecture of GoogleNet Inception-ResNet [11] and adapting the Inception block. On the one hand, it is scalable and can increase the receptive field of the network without increasing the number of parameters and computations. On the other hand, the multi-branch structure facilitates the extraction and fusion of features at different levels. The proposed network has 16 Inception blocks. The architecture of the Inception block and a lightweight version we propose in this paper are shown in Figure 3. In the Inception block, we use 1×1 convolution kernels to fuse and compress the channels, and use three-way branches to learn different residual features, and finally concatenate the three-way branches. We also design a lightweight Inception block, which will be denoted by (-) in what follows. With roughly the same number of parameters as a 3×3 Conv-BN-ReLU block for 64-layer feature maps, the proposed Inception block increases the network depth (3 non-linearities) and has a larger receptive field (5×5). Moreover, the Inception(-) uses about 50% of the parameters of the 3×3 Conv-BN-ReLU. The parameter comparison data are shown in Table 1.

3.2. Denoising after demosaicking

Since the demosaicking stage is trained in a noise-free setting, when a noisy input is demosaicked its output will contain a correlated residual noise. Removing this noise also requires learning. We therefore propose to use a denoising network to remove the structured noise resulting from demosaicking a noisy CFA image. We first learn a network for each noise level, but in the experiment sections we will also consider a noise level flexible network trained on a range of noise levels (with $\sigma \in [0, 20]$ as in [17]). In the noise level flexible network, the noise map shown in Figure 2 is introduced, which consists of the standard deviation σ of Gaussian noise added to the CFA image.

Table 1: Inception architecture and number of parameters. The depth of Conv-BN-ReLU is 1, the receptive field is 3, but the depth of the Inception is 3, and receptive field is 5. And Inception(-) has the same properties and the number of parameters is only 52.8% of Conv-BN-ReLU.

	Inception	Inception(-)	Conv.
Input the number of feature layers	64	64	64
First branch	$32(1 \times 1)$ $16(1 \times 1)$	$16(1 \times 1)$	3×3
Second branch	$32(1 \times 1)$ $32(3 \times 3)$ $16(3 \times 3)$	$16(1 \times 1)$ $16(3 \times 3)$	
Third branch	$32(1 \times 1)$ $32(3 \times 3)$ $32(3 \times 3)$	$16(1 \times 1)$ $32(3 \times 3)$ $32(3 \times 3)$	
Output the number of feature layers	64	64	64
Number of parameters	39360	19456	36992
GFLOPs (128×128)	0.649	0.321	0.607

For the denoising network, we also use the same Inception block architecture as the demosaicking network. As shown in Figure 2, the demosaicked image is used as input to the denoising stage. In addition, the features computed at the last layer of the demosaicking stage are reused by introducing them into the denoising stage by a skip connection.

3.3. Training procedure

The two stages of our method are trained independently, each with its own loss, which are both based on the classical mean square error (MSE) loss. In the first stage, the network is trained on a noise-free dataset. The loss for the noise-free demosaicking stage is

$$\mathcal{L}_{DM}(\Theta_{DM}) = \frac{1}{2N} \sum_{i=1}^{N} \left\| \widehat{X}_{DM}^{i} - X^{i} \right\|^{2}, \tag{7}$$

$$\widehat{X}_{DM}^{i} = IM. * \left(\widehat{X}_{GBTF}^{i} + F(\widehat{X}_{GBTF}^{i}; \Theta_{DM}) \right) + M. * \widehat{X}^{i}, \tag{8}$$

where $F(\widehat{X}_{GBTF}^{i}; \Theta_{DM})$ is the output of the demosaicking network to estimate the residual R_{GBTF} (see (4)).

After the demosaicking network is trained, we apply it to noisy CFA images (see model (1)) to produce noisy full-color images (see model (5)). The goal of the second stage is then to remove residual demosaicked noise ε_{DM}. Therefore the loss for this stage is

$$\mathcal{L}_{DN}(\Theta_{DN}) = \frac{1}{2N} \sum_{i=1}^{N} \left\| \widehat{X}_{DMDN}^{i} - X^{i} \right\|^{2}, \tag{9}$$

$$\widehat{X}^i_{DMDN} = \widehat{X}^i_{DM} - G(\widehat{X}^i_{DM}; \Theta_{DN}), \tag{10}$$

where $G(\widehat{X}^i_{DM}; \Theta_{DN})$ is the output of the denoising network, which works as an estimator of ε_{DM}.

For training the joint demosaicking and denoising, Gharbi *et al.* provided a dataset of two million 128×128 images (MIT Dataset) [17]. Ma *et al.* established the Waterloo Exploration Database (WED) with 4,744 high-quality natural images [61] and Syu *et al.* provided the Flickr500 with 500 high-quality images [48]. We use these datasets to build our training and test sets. Indeed, 100,000 images were randomly selected from the MIT dataset. And 4653 images in WED and 491 images in Flickr500 were randomly cropped into 100,000 images (128×128). These 200,000 patches (128×128) constitute our training set. Furthermore, 91 images in WED and 9 images in Flickr500 composed our test set. During the training time, the patch was flipped and rotated 180° with a 50% probability for data augmentation.

For training the denoising model we started by adding Gaussian white noise to the CFA images sampled from the training set (see Table 3 for the standard deviation σ of the noise) and applied the demosaicking network to the noisy CFA images. The color residual noise images, which were obtained by feeding the noisy CFA images into the demosaicing network, were utilized for training the denoising model.

The network architecture was implemented in Pytorch. The network weights were initialized using [62] and the biases were first set to 0. The optimization was performed by the ADAM optimizer [63] using the default parameters. The batch size was set to 64, and the initial learning rate to 10^{-2}. The learning rate decay strategy was the exponential decay method, and the learning rate decayed by 0.9 every 3000 iterations. Our model was trained on a NVIDIA Tesla V100 and required 50 epochs for each training iteration. The non-lightweight demosaicing and denoising algorithms at each level typically took approximately 3 days to train, while the lightweight algorithms could be trained within a day.

4. EXPERIMENTS

4.1. Datasets

We chose the classic Kodak [64] and McMaster [42] datasets for evaluating our algorithm on the demosaicking and denoising task. The Kodak dataset consists of 24 images (768×512). The McMaster dataset consists of 18 images (500×500), which were cropped from the 2310×1814 high-resolution images. At the same time, we conducted experiments on our test set, Urban100 dataset [65] and MIT moiré [17] to verify the reliability of our proposed algorithm. The Urban100 dataset is often used in super-resolution tasks and contains 100 high-resolution images. MIT moiré is the test set used by the JCNN algorithm [17], which contains 1000 images of 128×128 resolution that are prone to generate moiré.

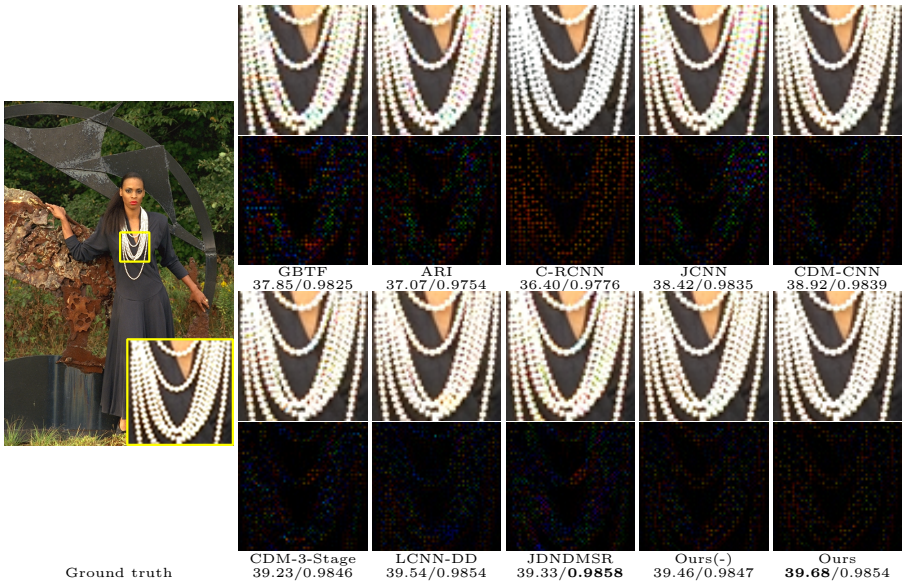


Figure 4: Results of the various comparisons between state-of-the-art and our method for noise-free demosaicking on image 18 of Kodak.

4.2. Quantitative and qualitative comparisons

Peak signal-to-noise ratio (PSNR) [66] and structural similarity (SSIM) [67] were used to evaluate the performance of the algorithms.

Noise-free demosaicking. In the noise-free CFA image demosaicking task, we compared three traditional algorithms (GBTF [31], MLRI+wei [35], ARI [36]) and six deep learning algorithms (C-RCNN [24], JCNN [17], CDM-CNN [18], CDM-3-Stage [46], LCNN-DD [59], JDNDMSR [25]). Table 2 summarizes the performance of all algorithms on the dataset. We can see that our proposed algorithm outperforms the other algorithms in the noise-free demosaicking. On the Kodak dataset, our proposed method surpasses the state-of-the-art by 0.34 dB in the average PSNR value. This gain is 0.27 dB on the McMaster dataset and 0.92 dB on Urban100. At the same time, our proposed lightweight method also achieved good performance. It ranks second on the Kodak and Urban100 dataset and third on the McMaster dataset. On the MIT moiré, the average PSNR value of our proposed algorithm is 0.12 dB lower than that of JCNN [17], but we only used 5% of the training data they provided.

Figure 4 illustrates a challenging case in which existing algorithms always produce color distortions (in the necklace part), while the proposed algorithm presents no distortion. In order to better observe the reconstruction effect of the algorithm, we show the residual image between the reconstructed image generated by all the algorithms and the ground truth. It can be seen that the visual effect is consistent with the numerical evaluation.

Table 2: Comparison with state-of-the-art algorithms in noise-free demosaicking. The best value is marked in **bold**, the second is marked in red, and the third is marked in blue. In the table, (-) indicates a lightweight version.

Algorithm	Kodak	McMaster	WED+Flickr	Urban100	MIT moiré	Average
	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
GBTF [31]	40.62/0.9859	34.38/0.9322	36.35/0.9664	34.82/0.9701	32.18/0.9120	35.67/0.9533
MLRI+wei [35]	40.26/0.9850	36.89/0.9620	36.76/0.9707	34.90/0.9732	32.17/0.9119	36.20/0.9606
ARI [36]	39.91/0.9815	37.57/0.9654	37.46/0.9745	35.35/0.9751	32.60/0.9162	36.58/0.9625
C-RCNN [24]	39.93/0.9843	36.68/0.9509	37.57/0.9725	37.16/0.9797	32.99/0.9148	36.87/0.9604
JCNN [17]	42.09/0.9881	38.95/0.9695	39.24/0.9807	38.12/0.9842	**36.65/0.9588**	39.01/0.9763
CDM-CNN [18]	41.98/0.9879	38.94/0.9696	39.52/0.9812	38.09/0.9836	34.28/0.9311	38.56/0.9707
CDM-3-Stage [46]	42.31/0.9885	39.34/0.9716	40.12/0.9827	38.60/0.9849	34.78/0.9334	39.03/0.9722
LCNN-DD [59]	42.42/0.9886	39.07/0.9701	39.75/0.9817	38.37/0.9841	34.78/0.9338	38.88/0.9717
JDNDMSR [25]	42.35/0.9891	38.83/0.9680	39.44/0.9810	38.33/0.9839	35.36/0.9338	38.86/0.9712
Ours(-)	42.49/0.9888	39.25/0.9702	39.84/0.9820	38.88/0.9852	35.97/0.9516	39.29/0.9756
Ours	**42.76/0.9893**	**39.61/0.9725**	**40.22/0.9831**	**39.52/0.9864**	36.53/0.9533	**39.73/0.9769**

Table 3: Comparison of the results (PSNR/SSIM) between different denoising and demosaicking methods for five image sets. The best value is marked in **bold**, the second is marked in red, and the third is marked in blue. The noise levels for which an algorithm doesn't work is indicated by "–" in the table. The algorithm noted by "*" which we don't obtain the source code, then the results (PSNR/SSIM) are taken from the article directly (the notation "x" in the table represents the unknown).

Algorithm		JCNN [17] PSNR/SSIM	C-RCNN [24] PSNR/SSIM	LCNN-DD [59] PSNR/SSIM	ADMM [56] PSNR/SSIM	SGNet*[20] PSNR/SSIM	JDNDMSR [25] PSNR/SSIM	1.5CBM3D [2] PSNR/SSIM	Ours(-) PSNR/SSIM	Ours PSNR/SSIM
Kodak										
σ	3	37.95/0.9626	37.25/0.9587	37.36/0.9428	31.85/0.8813	x	38.93/0.9678	38.73/0.9663	38.97/0.9677	**39.15/0.9685**
	5	36.18/0.9446	35.28/0.9327	33.91/0.8704	31.81/0.8765	x	36.99/0.9510	36.57/0.9482	37.01/0.9511	**37.12/0.9519**
	10	33.21/0.9007	30.94/0.8279	28.36/0.6710	31.22/0.8576	x	33.94/0.9115	33.34/0.9058	33.97//0.9124	**34.08/0.9143**
	15	31.32/0.8586	–	24.97/0.5201	30.30/0.8350	x	32.08/0.8752	31.44/0.8679	32.12/0.8780	**32.24/0.8807**
	20	29.91/0.8168	–	–	29.37/0.8115	–	30.79/0.8430	30.13/0.8343	30.89/0.8487	**31.01/0.8518**
	40	–	–	–	25.72/0.6797	–	–	26.88/0.7242	28.00/0.7621	**28.13/0.7663**
	60	–	–	–	24.22/0.6256	–	–	24.80/0.6533	26.46/0.7074	**26.58/0.7112**
McMaster										
σ	3	36.44/0.9470	34.36/0.9222	35.98/0.9254	32.51/0.9048	x	37.38/0.9546	37.15/0.9509	37.65/0.9554	**37.82/0.9567**
	5	35.31/0.9338	33.18/0.8985	33.29/0.8584	32.46/0.8985	x	36.14/0.9421	35.54/0.9347	36.29/0.9423	**36.41/0.9433**
	10	33.02/0.8972	29.66/0.7885	28.44/0.6740	31.64/0.8724	x	33.80/0.9126	32.84/0.8954	33.91/0.9132	**34.03/0.9152**
	15	31.25/0.8564	–	25.29/0.5309	30.46/0.8399	x	32.17/0.8858	31.03/0.8561	32.25/0.8867	**32.40/0.8902**
	20	29.79/0.8139	–	–	29.29/0.8068	–	30.93/0.8608	29.66/0.8186	31.06/0.8635	**31.21/0.8670**
	40	–	–	–	25.12/0.6650	–	–	25.90/0.6971	28.04/0.7902	**28.21/0.7962**
	60	–	–	–	22.92/0.5957	–	–	23.33/0.6155	26.28/0.7369	**26.45/0.7422**
WED + Flickr										
σ	3	36.28/0.9592	35.19/0.9486	35.92/0.9372	31.35/0.9060	x	37.32/0.9663	37.38/0.9646	37.72/0.9676	**37.92/0.9685**
	5	35.07/0.9448	33.77/0.9241	33.09/0.8660	31.32/0.9003	x	35.98/0.9542	35.69/0.9485	36.27/0.9554	**36.40/0.9563**
	10	32.70/0.9081	30.38/0.8393	28.15/0.6758	30.72/0.8808	x	33.56/0.9258	32.85/0.9108	33.76/0.9276	**33.88/0.9294**
	15	30.96/0.8719	–	24.98/0.5363	29.78/0.8580	x	31.93/0.8998	31.00/0.8778	32.07/0.9022	**32.20/0.9048**
	20	29.53/0.8354	–	–	28.78/0.8341	–	30.71/0.8762	29.63/0.8484	30.86/0.8800	**31.00/0.8832**
	40	–	–	–	24.94/0.7167	–	–	25.93/0.7523	27.84/0.8083	**27.98/0.8127**
	60	–	–	–	22.84/0.6625	–	–	23.40/0.6832	26.10/0.7565	**26.23/0.7613**
Urban 100										
σ	3	34.87/0.9680	34.98/0.9680	35.25/0.9560	28.53/0.9010	x	36.47/0.9749	36.70/0.9741	37.07/0.9768	**37.40/0.9779**
	5	33.69/0.9569	33.47/0.9526	32.72/0.9101	28.71/0.8987	34.54/0.9533	35.11/0.9665	34.89/0.9621	35.52/0.9683	**35.77/0.9694**
	10	31.26/0.9248	29.90/0.8895	27.97/0.7776	28.67/0.8864	32.14/0.9229	32.57/0.9438	31.86/0.9307	32.81/0.9459	**33.04/0.9482**
	15	29.45/0.8912	–	24.82/0.6692	28.08/0.8681	30.37/0.8923	30.79/0.9206	29.96/0.9018	30.96/0.9233	**31.21/0.9268**
	20	28.00/0.8552	–	–	27.26/0.8463	–	29.44/0.8972	28.58/0.8744	29.59/0.9012	**29.86/0.9060**
	40	–	–	–	23.60/0.7209	–	–	24.99/0.7726	26.17/0.8182	**26.47/0.8277**
	60	–	–	–	22.05/0.6628	–	–	22.56/0.6833	24.19/0.7476	**24.45/0.7582**
MIT moiré										
σ	3	33.66/0.9331	31.69/0.8853	33.08/0.9094	28.44/0.8296	x	33.97/0.9216	34.69/0.9360	34.84/0.9404	**35.28/0.9427**
	5	32.57/0.9172	30.68/0.8686	31.27/0.8699	28.48/0.8237	32.15/0.9043	32.84/0.9094	33.21/0.9145	33.52/0.9266	**33.81/0.9289**
	10	30.39/0.8724	28.08/0.8008	27.33/0.7465	28.19/0.8031	30.09/0.8619	30.73/0.8759	30.71/0.8701	31.19/0.8902	**31.40/0.8939**
	15	28.81/0.8283	–	24.46/0.6347	27.55/0.7801	28.60/0.8188	29.28/0.8427	29.12/0.8344	29.60/0.8544	**29.83/0.8597**
	20	27.57/0.7842	–	–	26.82/0.7564	–	28.21/0.8107	27.95/0.8019	28.47/0.8213	**28.71/0.8288**
	40	–	–	–	23.76/0.6387	–	–	24.96/0.6882	25.77/0.7176	**26.01/0.7279**
	60	–	–	–	22.43/0.5806	–	–	22.93/0.5946	24.25/0.6421	**24.46/0.6535**

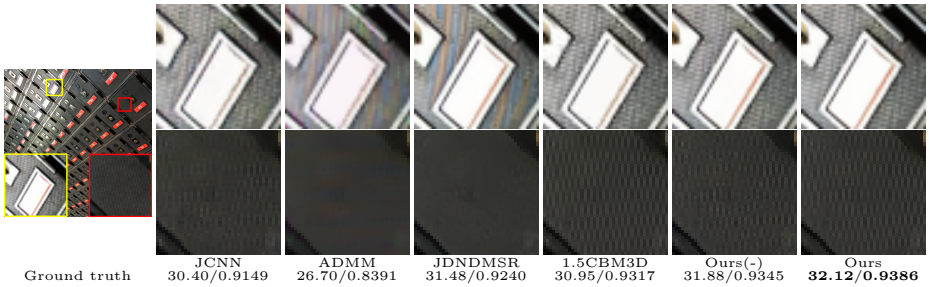


Figure 5: Comparison between state-of-the-art algorithms and our method for demosaicking and denoising in image 6 of the Urban dataset with noise $\sigma = 10$.

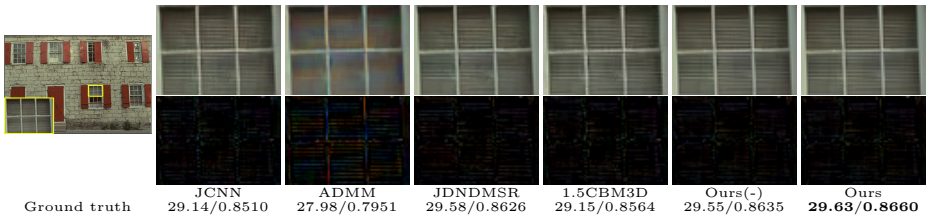


Figure 6: Comparison between state-of-the-art algorithms and our method for demosaicking and denoising in image 1 of the Kodak dataset with noise $\sigma = 15$.

Joint demosaicking and denoising. For the task of demosaicking and denoising of noisy CFA images, we compared with the joint demosaicking and denoising algorithm using ADMM by [56]. The joint demosaicking and denoising algorithms based on deep learning proposed in [17] (JCNN), in [24] (C-RCNN), in [59] (LCNN-DD), in [20] (SGNet[1]) and in [25] (JDNDMSR). We also considered our proposed demosaicking network combined with CBM3D for denoising [68], and following the suggestion of Jin *et al.*[2], the CBM3D denoising parameter was set to 1.5 times the original σ value (denoted 1.5CBM3D). Table 3 summarizes the performance comparison of all algorithms. It can be seen that our algorithm performs better than other state-of-the-art algorithms.

Figure 5-8 show the comparison of visual effects and image quality between the state-of-the-art and our proposed method. As can be seen in Figure 5 and 8, our restored images show a more distinct image texture and fine detail. Figure 6 illustrates that on the fence: our restored image is more pleasant and has fewer color distortions and checkerboard artifacts. We also note that CBM3D + our proposed demosaicking also outperforms the state-of-the-art for both quantitative and visual quality.

[1]Since we didn't obtain the source code of the algorithm SGNet [20], the PSNR and SSIM values of the algorithm were taken from the article directly.

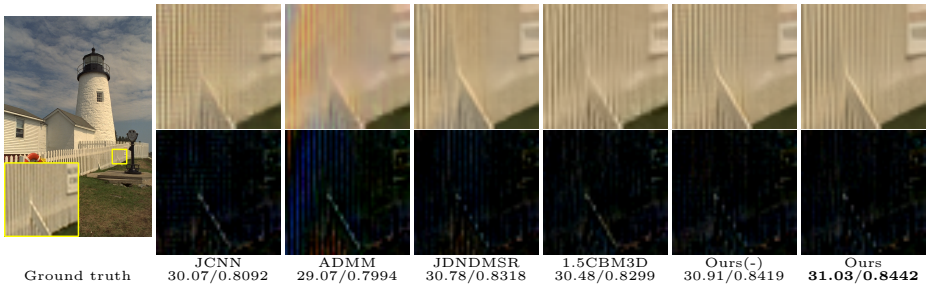


Figure 7: Comparison between state-of-the-art algorithms and our method for demosaicking and denoising in image 1 of the Kodak dataset with noise $\sigma = 20$.

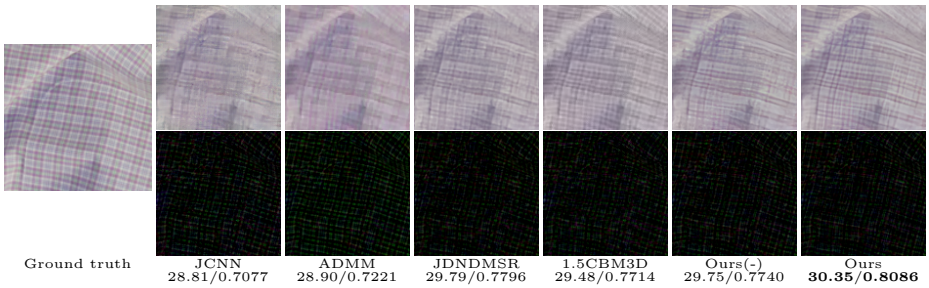


Figure 8: Comparison between state-of-the-art algorithms and our method for demosaicking and denoising in image 585 of the MIT moiré with noise $\sigma = 20$.

Table 4: Comparison of the results (PSNR/SSIM) between different flexible joint demosaicking and denoising methods in the interval of the noise level $\sigma \in (0, 20]$ for five image sets. The best value is marked in **bold**, the second is marked in red.

σ	Dataset	JCNN	JDNDMSR	Ours(-)-F	Ours-F
10	Kodak	33.21/0.9007	33.94/0.9115	33.92/0.9116	**34.03/0.9129**
	McMaster	33.02/0.8972	33.80/0.9126	33.85/0.9121	**33.97/0.9142**
	WED + Flickr	32.70/0.9081	33.56/0.9258	33.70/0.9269	**33.83/0.9284**
	Urban 100	31.26/0.9248	32.57/0.9438	32.75/0.9454	**32.95/0.9471**
	MIT moiré	30.39/0.8724	30.73/0.8759	31.09/0.8884	**31.31/0.8919**
20	Kodak	29.91/0.8168	30.79/0.8430	30.75/0.8441	**30.86/0.8465**
	McMaster	29.79/0.8139	30.93/0.8608	30.88/0.8571	**31.02/0.8609**
	WED + Flickr	29.53/0.8354	30.71/0.8762	30.71/0.8750	**30.84/0.8780**
	Urban 100	28.00/0.8552	29.44/0.8972	29.39/0.8963	**29.61/0.9002**
	MIT moiré	27.57/0.7842	28.21/0.8107	28.28/0.8141	**28.49/0.8207**

Noise level flexible joint demosaicking and denoising. Referring to [17, 20, 25], the noise level map was introduced in the denoising stage to flexibly handle the noise of a certain range of noise levels ($\sigma \in (0, 20]$). The corresponding PSNR and SSIM values are shown in Table 4. One can observe that the proposed method is superior to JCNN [17] and JDNDMSR [25] for all five image

Table 5: NIQE comparison between our proposed method and JCNN in DND dataset.

	JCNN	Ours(-)	Ours
linRGB	13.2701	8.8395	**4.3157**
sRGB	12.2354	7.8584	**3.8507**

databases. Our lightweight method is very competitive with JDNDMSR [25] and outperforms JCNN [17].

4.3. Results on real image datasets

Since the raw data is represented in the linear RGB color space (ie, without gamma transformation), inspired by [21, 69], we used the unprocessing algorithm [69] to convert the training data to linear RGB data and fine-tune the proposed algorithm. We evaluated the proposed algorithm on real images from the Darmstadt Noise Dataset (DND) [70]. Since there is no ground truth for these real world images, we decided to use the qualitative natural image quality evaluator (NIQE) [71] to evaluate the perceptual quality of the reconstructed images. The only input of NIQE is the restored image. Lower NIQE scores mean higher image quality. The NIQE scores of the restored results for real images were compared in linear RGB and sRGB color spaces for the various algorithms and are shown in Table 5. The NIQE scores of our method and of its lightweight version are much lower than that of JCNN. Figure 9 shows the restored images of the various algorithms in the sRGB space providing a visual quality confirmation of these measurements. A key part of each image in a red box is zoomed in and placed on the right side to make comparison easier. One can see that our proposed method restores better the textures and suppresses more noise than JCNN. Taking the first column of Figure 9 for example, the restored image of JCNN can't recover the top left curve, which is broken in the middle.

4.4. Ablation study and running time

Architecture choices, ablation study. Our ablation experiments trained and compared the following models:

- (a) Using GBTF [31] for preprocessing, while the demosaicking network is built using 16 classic Conv-BN-ReLU blocks (consistent with our network parameters).
- (b) Using GBTF [31] for preprocessing, while the demosaicking network is built using 8 Resblocks (consistent with our network parameters).
- (c) Using HA [27] for preprocessing, while the network uses our proposed Inception block.
- (d) Using bilinear interpolation for preprocessing, while the network uses our proposed Inception block.

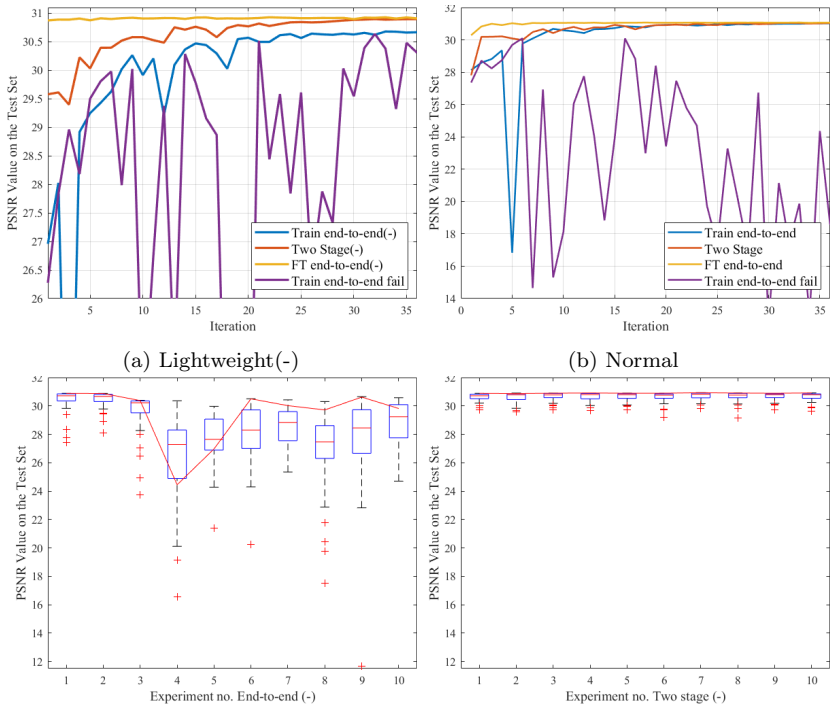
Figure 9: Comparison of JCNN with our method for demosaicking and denoising in real images of DND. Each group of images consists of the whole image and a part of the image. The image on the left is the whole image, and the image on the right is the zoomed in image of the part in the red box on the left.

The performance of the above four cases on the five datasets is shown in Table 6 (A). As can be seen from the table, good results can also be obtained using bilinear interpolation, but GBTF is a better choice when working with textured images. The table also shows that GBTF for preprocessing and using Inception blocks are more effective for image demosaicking.

Two-stage vs. end-to-end training. To verify the importance of the two-stage training we compared it with a joint demosaicking and denoising network trained end-to-end. For this experiment we set the noise level to $\sigma = 20$. Table 6 (B) shows the difference between both strategies. We can see that the end-to-end training of the networks (with equivalent capacity) is not as effective as the two-stage training. This highlights the importance of training first the demosaicking network on noise-free data. The network parameters from the two-stage training can actually be further refined with an end-to-end fine-tuning, which results in a slight boost. As can be seen from the training process in Figure 10, the two-stages training followed by fine-tuning allows for more stable training with better results. In our experiments, we also found that the two-stages training is more robust and more independent from initialization. On the contrary, end-to-end training is more sensitive to initial values and is prone to training failure. In Figure 10 (a) and (b), we list the training records of an end-to-end model that failed to train once. As shown in Figure 10 (a) and (b), the end-to-end training is prone to failure due to training fluctuations, while two-stage training results in a smooth progression. To compare the training robustness of the different schemes, we trained the lightweight end-to-end network and the two-stage network 10 times respectively. The training results are shown in Figure 10 (c). One can see that the end-to-end network is not stable. Its final results are not the highest value seen during the training process. There is only a 20% success rate, while the two-stage method is very stable and the final result

Table 6: Ablation study. Sub-table A is the choice of network structure. Sub-table B shows the comparison of two-stage training with end-to-end training.

Method	Kodak PSNR/SSIM	McMaster PSNR/SSIM	WED+Flickr PSNR/SSIM	Urban100 PSNR/SSIM	MIT moiré PSNR/SSIM	Average PSNR/SSIM
A. Demosaic						
GBTF+Conv	42.38/0.9886	39.28/0.9707	39.79/0.9818	38.78/0.9851	35.78/0.9503	39.20/0.9753
GBTF+Resblock	42.34/0.9884	39.32/0.9712	39.83/0.9820	38.81/0.9852	35.78/0.9501	39.22/0.9754
HA+Inception	42.14/0.9877	39.28/0.9707	39.71/0.9816	38.62/0.9849	35.72/0.9501	39.09/0.9750
Billinear+Inception	42.60/0.9889	**39.62/0.9727**	40.17/0.9830	39.21/0.9862	36.35/0.9536	39.59/0.9769
Ours(-)	42.49/0.9888	39.25/0.9702	39.84/0.9820	38.88/0.9852	35.97/0.9516	39.29/0.9756
Ours	**42.76/0.9893**	39.61/0.9725	**40.22/0.9831**	**39.52/0.9864**	**36.53/0.9533**	**39.73/0.9769**
B. End-to-End Joint demosaicking and Denosing ($\sigma = 20$)						
End-to-end training(-)	30.71/0.8436	30.87/0.8587	30.66/0.8750	29.20/0.8937	28.11/0.8090	29.91/0.8560
Two-Stage training(-)	30.89/0.8487	31.06/0.8635	30.86/**0.8800**	29.59/**0.9012**	28.47/0.8213	30.17/**0.8629**
End-to-end FT of Two-stage(-)	**30.90/0.8489**	**31.08/0.8637**	**30.87**/0.8796	**29.61**/0.9008	**28.49/0.8214**	**30.19/0.8629**
End-to-end training	30.99/0.8514	31.17/0.8664	30.97/0.8828	29.81/0.9051	28.61/0.8263	30.31/0.8664
Two-Stage training	31.01/0.8518	31.21/0.8670	**31.00/0.8832**	29.86/**0.9060**	28.71/0.8288	30.36/0.8674
End-to-end FT of Two-stage	**31.02/0.8520**	**31.22/0.8671**	**31.00**/0.8829	**29.87**/0.9059	**28.73/0.8294**	**30.37/0.8675**



(a) Lightweight(-) (b) Normal

(c) Training ten times each

Figure 10: Plots (a) and (b) compare the performance of different training strategies along the training iterations. We compare the end-to-end training, the two-stage training and finetuning after the two-stage training. We also report the evolution of a failed end-to-end training (purple curve), which were obtained with the same parameters as the blue curve. In (c), the end-to-end network and the two-stage network were each trained 10 times (for this experiment we used only the lightweight architecture). The unstable behavior (as in the purple curve) was observed in eight out of ten end-to-end trainings, while the the two stage training never exhibited such behavior. The red curve marks the PSNR reached at the end of each training.

always reaches the highest value of each training. This shows that, although CNNs have a powerful fitting capability that enables addressing multiple tasks in an end-to-end fashion, it is still important to consider the order of the tasks to design a reasonable pipeline.

Dependency on the training dataset. In order to better compare the advantages of the network architecture regardless of the influence of training data and training strategies, we retrained JCNN, using the same training data and training strategy as for our algorithm. Table 7 shows the PSNR and SSIM of noise-free demosaicking and joint demosaicking and denoising with noise level $\sigma = 20$. Among them, JCNN-O represents the original parameters of JCNN and JCNN-R stands for the retrained version of JCNN. From Table 7, one can see that both our proposed method and its lightweight version outperform JCNN by a

Table 7: Comparison of the results (PSNR/SSIM) of original JCNN (JCNN-O), retrained JCNN (JCNN-R) and the proposed method for five image sets. The best value is marked in **bold**, the second is marked in red.

σ	Dataset	JCNN-O	JCNN-R	Ours(-)	Ours
0	Kodak	42.09/0.9881	41.65/0.9874	42.49/0.9888	**42.76/0.9893**
	McMaster	38.95/0.9695	38.68/0.9677	39.25/0.9702	**39.61/0.9725**
	WED + Flickr	39.24/0.9807	39.11/0.9800	39.84/0.9820	**40.22/0.9831**
	Urban 100	38.12/0.9842	37.97/0.9833	38.88/0.9852	**39.52/0.9864**
	MIT moiré	**36.65/0.9588**	35.19/0.9462	35.97/0.9516	36.53/0.9533
15	Kodak	31.32/0.8586	31.37/0.8597	32.12/0.8780	**32.24/0.8807**
	McMaster	31.25/0.8564	31.31/0.8645	32.25/0.8867	**32.40/0.8902**
	WED + Flickr	30.96/0.8719	31.09/0.8819	32.07/0.9022	**32.20/0.9048**
	Urban 100	29.45/0.8912	29.18/0.8953	30.96/0.9233	**31.21/0.9268**
	MIT moiré	28.81/0.8283	28.36/0.8165	29.60/0.8544	**29.83/0.8597**
20	Kodak	29.91/0.8168	30.04/0.8228	30.89/0.8487	**31.01/0.8518**
	McMaster	29.79/0.8139	30.08/0.8338	31.06/0.8635	**31.21/0.8670**
	WED + Flickr	29.53/0.8354	29.83/0.8529	30.86/0.8800	**31.00/0.8832**
	Urban 100	28.00/0.8552	27.82/0.8642	29.59/0.9012	**29.86/0.9060**
	MIT moiré	27.57/0.7842	27.27/0.7758	28.47/0.8213	**28.71/0.8288**

Table 8: Average running time of demosaicking and joint demosaicking-denoising for 500 images (512×512) on a PC with Intel Core i7-9750H 2.60GHz, 16GB memory, and Nvidia GTX-1650 GPU.

	Method	CPU(s)	GPU(s)	GFLOPs	Para(M)
DM	GBTF [31]	2.74	–	–	–
	MLRI+wei [35]	1.35	–	–	–
	ARI [36]	25.58	–	–	–
	CDM-CNN [18]	6.84	0.07	276.19	0.53
	CDM-3-Stage [46]	18.61	0.35	1871.27	3.57
	Ours(-) (DM)	15.12	0.28	92.34	0.35
	Ours (DM)	24.86	0.44	176.23	0.67
JDD	ADMM [56]	472.27	–	–	–
	C-RCNN [24]	112.77	2.32	2112.8	0.38
	JCNN [17]	10.41	0.22	53.20	0.56
	LCNN-DD [59]	1.86	0.04	14.89	0.23
	JDNDMSR [25]	73.15	1.61	1641.77	6.33
	Ours(-) (DM+DN)	30.13	0.55	184.68	0.70
	Ours (DM+DN)	49.53	0.86	352.46	1.34

margin larger than 0.7 dB for all five test image sets under the same training data and training strategy. This means that our proposed structure is superior to JCNN for both tasks.

Computational complexity. In order to estimate the computational complexity of these algorithms, we tested the average time consumed by all algorithms to process 500 images (512×512) on a PC with Intel Core i7-9750H 2.60GHz, 16GB memory, and Nvidia GTX-1650 GPU. For the deep learning algorithms, only the actual network processing time was calculated, not including the network loading time. The time consumed by the algorithm in demosaicking noise-free

images and in demosaicking and denoising CFA images with noise level of $\sigma = 10$ is shown in Table 8. Since our network is composed of independent demosaicking and denoising stages, the time consumed can be calculated separately. In Table 8, DM denotes the demosaicking stage of our algorithm and JDD denotes joint demosaicking and denoising. It can be seen that the processing time of our algorithm is comparable to the other deep learning algorithms. It is also faster than some traditional iterative algorithms, such as ARI [36] and ADMM [56].

5. Conclusion

In this paper, we proposed a CNN for joint demosaicking and denoising. The proposed method relies on a demosaicking first then denoising approach, which is realized by applying sequentially two CNNs. In the first stage, the GBTF algorithm is combined with a CNN to reconstruct a full-color image from noisy CFA image but ignoring the image noise. In the second stage, we use another CNN to learn to remove the noise whose statistical properties were changed by the demosaicking stage. This allows to remove demosaicing noise that would otherwise be virtually impossible to remove using model-based methods.

More importantly, we show that even when dealing with CNNs with powerful fitting capabilities a reasonable pipeline and its training (such as the proposed two-stage training) can lead to significant performance gains with respect to more mainstream approaches based on end-to-end training. In addition, in order to improve the performance of the proposed method, we proposed an architecture based on Inception blocks as well as a lightweight version with a good speed-performance trade-off. Experiments conducted on multiple datasets confirmed that our algorithm favourably compares to the state-of-the-art demosaicking algorithms and joint demosaicking and denoising algorithms.

Acknowledgment

This work was supported by National Natural Science Foundation of China (No. 12061052), Young Talents of Science and Technology in Universities of Inner Mongolia Autonomous Region (No. NJYT22090), Natural Science Fund of Inner Mongolia Autonomous Region (No. 2020MS01002), Innovative Research Team in Universities of Inner Mongolia Autonomous Region (No. NMGIRT2207), Special Funds for Graduate Innovation and Entrepreneurship of Inner Mongolia University (No. 11200-121024), Prof. Guoqing Chen's "111 project" of higher education talent training in Inner Mongolia Autonomous Region and the network information center of Inner Mongolia University. Work partly financed by Office of Naval research grants N00014-17-1-2552 and N00014-20-S-B001, DGA Defals challenge n° ANR-16-DEFA-0004-01, MENRT and Fondation Mathématique Jacques Hadamard. Y. Guo and Q. Jin are very grateful to Professor Guoqing Chen for helpful comments and suggestions. The authors are also grateful to the reviewers for their valuable comments and remarks.

References

[1] B. E. Bayer, Color imaging array, 1976. US Patent 3,971,065.

[2] Q. Jin, G. Facciolo, J. Morel, A review of an old dilemma: Demosaicking first, or denoising first?, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recogn. Workshops, 2020, pp. 2169–2179. doi:10.1109/CVPRW50498.2020.00265.

[3] O. Kalevo, H. Rantanen, Noise reduction techniques for bayer-matrix images, in: Proc. Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications III, volume 4669, International Society for Optics and Photonics, 2002, pp. 348–359.

[4] S. H. Park, H. S. Kim, S. Lansel, M. Parmar, B. A. Wandell, A case for denoising before demosaicking color filter array data, in: Proc. Conf. Rec. Asilomar Conf. Signals Syst. Comput., 2009, pp. 860–864. doi:10.1109/ACSSC.2009.5469990.

[5] M. Lee, S. Park, M. Kang, Denoising algorithm for cfa image sensors considering inter-channel correlation, Sensors 17 (2017) 1236.

[6] L. Condat, A simple, fast and efficient approach to denoisaicking: Joint demosaicking and denoising, in: Proc. IEEE Int. Conf. Image Process., 2010, pp. 905–908. doi:10.1109/ICIP.2010.5652196.

[7] L. Condat, S. Mosaddegh, Joint demosaicking and denoising by total variation minimization, in: Proc. IEEE Int. Conf. Image Process., 2012, pp. 2781–2784. doi:10.1109/ICIP.2012.6467476.

[8] L. Condat, A generic proximal algorithm for convex optimization—application to total variation minimization, IEEE Signal Process. Lett. 21 (2014) 985–989. doi:10.1109/LSP.2014.2322123.

[9] A. Danielyan, M. Vehvilainen, A. Foi, V. Katkovnik, K. Egiazarian, Cross-color bm3d filtering of noisy raw data, in: Proc. Int. Workshop Local Non-Local Approx. Image Process., 2009, pp. 125–129. doi:10.1109/LNLA.2009.5278395.

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

[11] C. Szegedy, S. Ioffe, V. Vanhoucke, A. A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: Thirty-first AAAI conference on artificial intelligence, 2017, p. 4278–4284.

[12] K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising, IEEE Trans. Image Process. 26 (2017) 3142–3155.

[13] K. Zhang, W. Zuo, L. Zhang, Ffdnet: Toward a fast and flexible solution for cnn-based image denoising, IEEE Trans. Image Process. 27 (2018) 4608–4622. doi:10.1109/TIP.2018.2839891.

[14] Y. Guo, A. Davy, G. Facciolo, J.-M. Morel, Q. Jin, Fast, nonlocal and neural: A lightweight high quality solution to image denoising, IEEE Signal Process. Lett. 28 (2021) 1515–1519. doi:10.1109/LSP.2021.3099963.

[15] F. Fang, J. Li, Y. Yuan, T. Zeng, G. Zhang, Multilevel edge features guided network for image denoising, IEEE Transactions on Neural Networks and Learning Systems 32 (2021) 3956–3970. doi:10.1109/TNNLS.2020.3016321.

[16] R. Hou, F. Li, Idpcnn: Iterative denoising and projecting cnn for mri reconstruction, Journal of Computational and Applied Mathematics 406 (2022) 113973. doi:https://doi.org/10.1016/j.cam.2021.113973.

[17] M. Gharbi, G. Chaurasia, S. Paris, F. Durand, Deep joint demosaicking and denoising, ACM Trans. Graph. 35 (2016) 191.

[18] R. Tan, K. Zhang, W. Zuo, L. Zhang, Color image demosaicking via deep residual learning, in: Proc. IEEE Int. Conf. Multimedia Expo (ICME), 2017, pp. 793–798.

[19] D. S. Tan, W. Chen, K. Hua, Deepdemosaicking: Adaptive image demosaicking via multiple deep fully convolutional networks, IEEE Trans. Image Process. 27 (2018) 2408–2419.

[20] L. Liu, X. Jia, J. Liu, Q. Tian, Joint demosaicing and denoising with self guidance, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 2237–2246.

[21] S. Guo, Z. Liang, L. Zhang, Joint denoising and demosaicking with green channel prior for real-world burst images, IEEE Trans. Image Process. 30 (2021) 6930–6942. doi:10.1109/TIP.2021.3100312.

[22] F. Fang, J. Li, T. Zeng, Soft-edge assisted network for single image super-resolution, IEEE Trans. Image Process. 29 (2020) 4656–4668.

[23] Z. Wen, J. Guan, T. Zeng, Y. Li, Residual network with detail perception loss for single image super-resolution, Computer Vision and Image Understanding 199 (2020) 103007.

[24] F. Kokkinos, S. Lefkimmiatis, Deep image demosaicking using a cascade of convolutional residual denoising networks, in: Proc. Eur. Conf. Comput. Vis., 2018, pp. 303–319.

[25] W. Xing, K. Egiazarian, End-to-end learning for joint image demosaicing, denoising and super-resolution, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 3507–3516.

[26] C. A. Laroche, M. A. Prescott, Apparatus and method for adaptively interpolating a full color image utilizing chrominance gradients, 1994. US Patent 5,373,322.

[27] J. F. Hamilton Jr, J. E. Adams Jr, Adaptive color plan interpolation in single sensor color electronic camera, 1997. US Patent 5,629,734.

[28] J. E. Adams, Design of practical color filter array interpolation algorithms for digital cameras .2, in: Proc. IEEE Int. Conf. Image Process., volume 1, 1998, pp. 488–492 vol.1.

[29] Q. Jin, Y. Guo, J.-M. Morel, G. Facciolo, A Mathematical Analysis and Implementation of Residual Interpolation Demosaicking Algorithms, Image Processing On Line 11 (2021) 234–283. https://doi.org/10.5201/ipol.2021.358.

[30] Lei Zhang, Xiaolin Wu, Color demosaicking via directional linear minimum mean square-error estimation, IEEE Trans. Image Process. 14 (2005) 2167–2178.

[31] I. Pekkucuksen, Y. Altunbasak, Gradient based threshold free color filter array interpolation, in: Proc. IEEE Int. Conf. Image Process., 2010, pp. 137–140.

[32] D. Kiku, Y. Monno, M. Tanaka, M. Okutomi, Residual interpolation for color image demosaicking, in: Proc. IEEE Int. Conf. Image Process., 2013, pp. 2304–2308.

[33] K. He, J. Sun, X. Tang, Guided image filtering, IEEE Trans. Pattern Anal. Mach. Intell. 35 (2013) 1397–1409.

[34] D. Kiku, Y. Monno, M. Tanaka, M. Okutomi, Minimized-laplacian residual interpolation for color image demosaicking, in: SPIE, volume 9023, 2014, pp. 90230L–1–90230L–8.

[35] D. Kiku, Y. Monno, M. Tanaka, M. Okutomi, Beyond color difference: Residual interpolation for color image demosaicking, IEEE Trans. Image Process. 25 (2016) 1288–1300.

[36] Y. Monno, D. Kiku, M. Tanaka, M. Okutomi, Adaptive residual interpolation for color and multispectral image demosaicking, Sensors 17 (2017) 2787.

[37] A. Buades, B. Coll, J. Morel, C. Sbert, Self-similarity driven color demosaicking, IEEE Trans. Image Process. 18 (2009) 1192–1202.

[38] J. Duran, A. Buades, Self-similarity and spectral correlation adaptive algorithm for color demosaicking, IEEE Trans. Image Process. 23 (2014) 4031–4040.

[39] J. Mairal, F. Bach, J. Ponce, G. Sapiro, A. Zisserman, Non-local sparse models for image restoration, in: Proc. IEEE Int. Conf. Comput. Vis., 2009, pp. 2272–2279.

[40] Y. M. Lu, M. Karzand, M. Vetterli, Demosaicking by alternating projections: Theory and fast one-step implementation, IEEE Trans. Image Process. 19 (2010) 2085–2098.

[41] J. Zhang, A. Sheng, K. Hirakawa, A wavelet-gsm approach to demosaicking, IEEE Signal Process. Lett. 25 (2018) 778–782.

[42] E. Dubois, Frequency-domain methods for demosaicking of bayer-sampled color images, IEEE Signal Process. Lett. 12 (2005) 847–850.

[43] E. Dubois, Filter design for adaptive frequency-domain bayer demosaicking, in: Proc. Int. Conf. Image Process., 2006, pp. 2705–2708.

[44] K.-L. Hua, S. C. Hidayati, F.-L. He, C.-P. Wei, Y.-C. F. Wang, Context-aware joint dictionary learning for color image demosaicking, Journal of Visual Communication and Image Representation 38 (2016) 230–245.

[45] C. Bai, J. Li, Z. Lin, Demosaicking based on channel-correlation adaptive dictionary learning, Journal of Electronic Imaging 27 (2018) 043047.

[46] K. Cui, Z. Jin, E. Steinbach, Color image demosaicking using a 3-stage convolutional neural network structure, in: Proc. IEEE Int. Conf. Image Process., 2018, pp. 2177–2181.

[47] H. Malvar, L. wei He, R. Cutler, High-quality linear interpolation for demosaicing of bayer-patterned color images, in: 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing, volume 3, 2004, pp. iii–485. doi:10.1109/ICASSP.2004.1326587.

[48] N.-S. Syu, Y.-S. Chen, Y.-Y. Chuang, Learning deep convolutional networks for demosaicing, arXiv:1802.03769 (2018).

[49] T. Yamaguchi, M. Ikehara, Image demosaicking via chrominance images with parallel convolutional neural networks, in: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 1702–1706.

[50] K. Mei, J. Li, J. Zhang, H. Wu, J. Li, R. Huang, Higher-resolution network for image demosaicing and enhancing, in: Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop, 2019, pp. 3441–3448.

[51] L. Condat, S. Mosaddegh, Joint demosaicking and denoising by total variation minimization, in: Proc. IEEE Int. Conf. Image Process., 2012, pp. 2781–2784.

[52] T. Klatzer, K. Hammernik, P. Knobelreiter, T. Pock, Learning joint demosaicing and denoising based on sequential energy minimization, in: Proc. IEEE Int. Conf. Comput. Photogr., 2016, pp. 1–11.

[53] D. Khashabi, S. Nowozin, J. Jancsary, A. W. Fitzgibbon, Joint demosaicing and denoising via learned nonparametric random fields, IEEE Trans. Image Process. 23 (2014) 4968–4981.

[54] D. Menon, G. Calvagno, Joint demosaicking and denoisingwith space-varying filters, in: Proc. IEEE Int. Conf. Image Process., 2009, pp. 477–480.

[55] D. Menon, G. Calvagno, Regularization approaches to demosaicking, IEEE Transactions on Image Processing 18 (2009) 2209–2220. doi:10.1109/TIP.2009.2025092.

[56] H. Tan, X. Zeng, S. Lai, Y. Liu, M. Zhang, Joint demosaicing and denoising of noisy bayer images with admm, in: Proc. IEEE Int. Conf. Image Process., 2017, pp. 2951–2955.

[57] S. Lefkimmiatis, Universal denoising networks : A novel cnn architecture for image denoising, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 3204–3213. doi:10.1109/CVPR.2018.00338.

[58] F. Kokkinos, S. Lefkimmiatis, Iterative joint image demosaicking and denoising using a residual denoising network, IEEE Trans. Image Process. 28 (2019) 4177–4188.

[59] T. Huang, F. F. Wu, W. Dong, G. Shi, X. Li, Lightweight deep residue learning for joint color image demosaicking and denoising, in: Proc. Int. Conf. Pattern Recognit., 2018, pp. 127–132.

[60] T. Ehret, A. Davy, P. Arias, G. Facciolo, Joint demosaicking and denoising by fine-tuning of bursts of raw images, in: Proc. IEEE/CVF Int. Conf. Comput. Vis., 2019, pp. 8867–8876.

[61] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, L. Zhang, Waterloo exploration database: New challenges for image quality assessment models, IEEE Trans. Image Process. 26 (2017) 1004–1016.

[62] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 1026–1034.

[63] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980 (2014).

[64] L. Zhang, X. Wu, A. Buades, X. Li, Color demosaicking by local directional interpolation and nonlocal adaptive thresholding, Journal of Electronic imaging 20 (2011) 023016.

[65] J. Huang, A. Singh, N. Ahuja, Single image super-resolution from transformed self-exemplars, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 5197–5206.

[66] D. Alleysson, S. Susstrunk, J. Herault, Linear demosaicing inspired by the human visual system, IEEE Trans. Image Process. 14 (2005) 439–449.

[67] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (2004) 600–612.

[68] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian, Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space, in: Proc. IEEE Int. Conf. Image Process., volume 1, 2007, pp. I – 313–I – 316.

[69] T. Brooks, B. Mildenhall, T. Xue, J. Chen, D. Sharlet, J. T. Barron, Unprocessing images for learned raw denoising, in: Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 11028–11037.

[70] T. Plötz, S. Roth, Benchmarking denoising algorithms with real photographs, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 2750–2759. doi:10.1109/CVPR.2017.294.

[71] A. Mittal, R. Soundararajan, A. C. Bovik, Making a "completely blind" image quality analyzer, IEEE Signal Process. Lett. 20 (2013) 209–212.