EUROGRAPHICS 2018 / D. Gutierrez and A. Sheffer
(Guest Editors)

Volume 37 (2018), Number 2

Deep Joint Design of Color Filter Arrays and Demosaicing

Bernardo Henz[1,2] Eduardo S. L. Gastal[1] Manuel M. Oliveira[1]

[1]Instituto de Informática – UFRGS
[2]Instituto Federal Farroupilha – IFFar

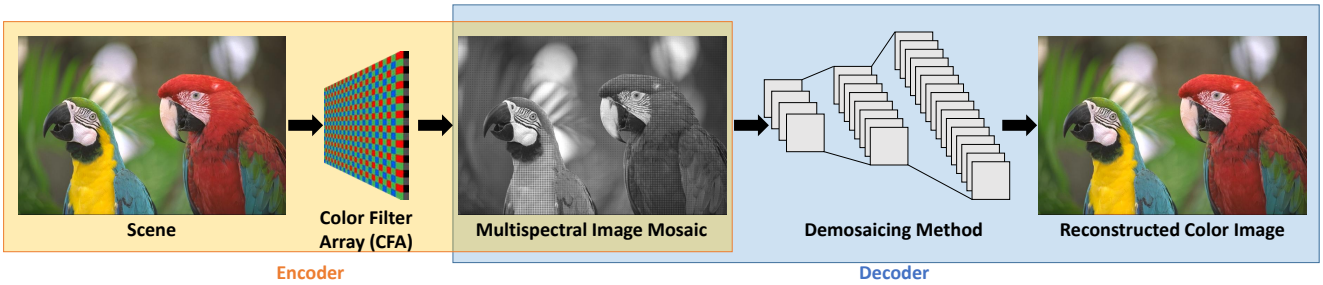


Figure 1: *Capture and reconstruction of color images on single-sensor cameras. A color filter array (CFA) selectively allows scene photons with certain wavelengths to reach portions of a monochromatic sensor. A color image is then reconstructed from the filtered samples (multispectral image mosaic) using a demosaicing algorithm. We model this process as an autoencoder: the CFA projection encodes color information onto the monochromatic sensor, which is later decoded by the color-reconstruction method. The joint design of CFA patterns and demosaicing produces high-quality color reconstruction, outperforming existing techniques.*

Abstract

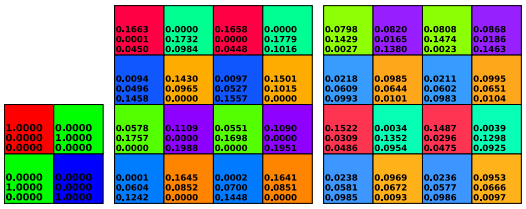
We present a convolutional neural network architecture for performing joint design of color filter array (CFA) patterns and demosaicing. Our generic model allows the training of CFAs of arbitrary sizes, optimizing each color filter over the entire RGB color space. The patterns and algorithms produced by our method provide high-quality color reconstructions. We demonstrate the effectiveness of our approach by showing that its results achieve higher PSNR than the ones obtained with state-of-the-art techniques on all standard demosaicing datasets, both for noise-free and noisy scenarios. Our method can also be used to obtain demosaicing strategies for pre-defined CFAs, such as the Bayer pattern, for which our results also surpass even the demosaicing algorithms specifically designed for such a pattern.

CCS Concepts

•***Computing methodologies → Computational photography; Neural networks;***

1. Introduction

Color filter arrays (CFAs) are a key component of digital imaging devices, allowing the capture of color pictures using a single monochromatic sensor. Superimposed on the sensor, a CFA selectively allows photons with certain wavelengths to reach the sensor, creating a *multispectral mosaic* (Fig. 1 (center)) defined by the properties of the filters in the CFA pattern. These filtered samples are then interpolated through a process known as *demosaicing* that reconstructs the resulting color image (Fig. 1 (right)). The Bayer pattern [Bay76] is the most popular CFA pattern for digital cameras and many demosaicing algorithms have been proposed to improve the quality of the reconstructed images sampled with it [MBP*09, Get11b, HvLP12, Wan14]. Nonetheless, several techniques have focused on designing new CFA patterns, either by looking for patterns whose representation in the frequency domain have little overlap between luma and chroma information [HLLD11, Con11, BLLY16] (designed specifically for use with frequency-selection demosaicing algorithms [ASH05, Dub05]), or by designing CFAs for sparse-representation-based demosaicings [LBLY17, MBP*09]. Essentially, previous solutions have been restricted to either design demosaicing algorithms for a given CFA pattern (*e.g.*, the Bayer pattern), or to design CFA



Bayer Our 4x4 noise-free Our 4x4 noise

Figure 2: *CFA patterns. From left to right: Bayer pattern, our* 4×4 *CFA for noise-free data, and our* 4×4 *CFA for noisy data. The numbers inside each cell are (from top to bottom) the R, G, and B coefficients defining the color filter.*

patterns that work with a specific demosaicing algorithm (*e.g.*, [ASH05, Dub05, MBP*09]). Since sampling and reconstruction are tightly-coupled processes, by predefining the CFA pattern or the reconstruction algorithm, one severely constrains the search space and, therefore, the ability to obtain an optimal solution.

We simultaneously address the problem of CFA pattern design and demosaicing. For this, we use an end-to-end autoencoder that mimics the process of image acquisition (Fig. 1). An autoencoder is a learning model that tries to reconstruct the original information after projecting it into a lower-dimensional space [KW13]. *Our autoencoder jointly obtains a CFA pattern and a demosaicing algorithm.* The encoding step projects the light filtered by the CFA onto a monochromatic sensor, generating a multispectral image mosaic. The decoding step recovers a color image from the mosaic (Fig. 1). By training encoder and decoder simultaneously, for any given CFA dimensions our approach automatically finds the CFA pattern and corresponding demosaicing algorithm that minimizes color-reconstruction error. Our technique produces high-quality color reconstructions, outperforming the state-of-the-art techniques in all standard demosaicing datasets both for noise-free and noisy data. Fig. 2 shows two 4×4 color filters obtained with our method. Such patterns are tiled to produce CFAs with the desired number of pixels.

The **contributions** of our work include:

- *A method for the joint design of CFA pattern and demosaicing that minimizes color-reconstruction errors* (Section 3). Our model is the first to optimize CFA colors over the entire RGB color space, while jointly optimizing demosaicing. The results produced by our system outperform existing solutions in terms of PSNR for both noise-free and noisy data (Section 4);
- *An autoencoder architecture that models the color-image capture process on single monochromatic sensors.* Our architecture achieves fast training convergence on image patches, and works with CFAs of different sizes, including existing ones (Section 3.1);

2. Related work

2.1. CFA Design

Following the work of Bayer [Bay76], several color filter arrays have been proposed over the years. Lukac and Plataniotis [LP05] analysed the performance of ten RGB CFAs. More recently, researchers have proposed a variety of new CFA design strategies. Hirakawa and Wolfe [HW08] introduced the idea of designing CFAs directly in the Fourier domain by optimizing the carrier waves. Lu and Vetterli [LV09] presented a CFA pattern that minimizes the reconstruction error of a linear-minimum-mean-square-error demosaicing method. Condat presented three different designs: RGB CFAs arranged in a non-periodic pattern [Con09], randomly generated patterns with specific blue-noise characteristics [Con10], and a 2×3 CFA pattern which has enhanced sensitivity properties and robustness to noise [Con11]. Hao et al. [HLLD11] introduced CFAs based on the frequency structure, which are manually initialized according to some guidelines, and then optimized using a geometric method. Bai et al. [BLLY16] proposed a method for CFA design in the frequency domain. Given the CFA pattern size, their method suggests frequency-structure candidates and then optimizes the parameters by maximizing numeric stability of color transformations. Chakrabarti et al. [CFZ14] proposed a predominantly panchromatic CFA that samples color at sparse sets of locations, which are then propagated throughout the image guided by the luminance channel. In a subsequent work, Chakrabarti [Cha16] used a convolutional neural network (CNN) architecture to design a CFA from a set of predefined colors, while training the demosaicing method concurrently. Li et al. [LBLY17] proposed a CFA design optimized for sparse-representation-based demosaicing [MBP*09] and showed how to minimize mutual coherence of CFAs with constraints for physical realizability.

The work closest related to ours is the one proposed by Chakrabarti [Cha16], which is the only previous technique to jointly optimize CFA design and demosaicing. However, our method differs from Chakrabarti's in a crucial aspect: while Chakrabarti treats CFA design as a decision problem, we model it as a regression problem, thus optimizing CFA colors over the entire RGB color space. As such, our approach explores a much larger parameter space, resulting in better color-reconstructed images, with PSNR gains ranging from 5 up to 30 dB, for images in the standard demosaicing datasets (Section 4).

2.2. Demosaicing

Demosaicing is a well-studied problem, with many surveys on existing methods [LGZ08, MC11, KB15]. Demosaicing algorithms have been proposed for the frequency domain [ASH05, Dub05, LCTZ07], and based on hard-coded heuristics for interpolation [JAJ*14, Li05, ZW05], self-similarities [ZWBL11a, BCMS09], optimization schemes [CM12, HST*14, KHKP16], and compressive sensing [MBP*09, MAKR13, DVM16]. Next, we discuss demosaicing strategies based on neural networks.

Kappa and Hel-Or [KHO00] and Go et al. [GSL00] were the first to use neural networks for demosaicing. Long and Huang [LH06] later proposed an adaptive scheme to improve Go et al.'s method. Heinze et al. [HvLP12] proposed multi-frame demosaicing using a neural network for estimating the pixel color based on its surroundings. Wang [Wan14] used 4×4 patches to train a multilayer neural network while minimizing a suitable objective function. Gharbi et al. constructed a dataset with hard cases, which were used to train a CNN for joint demosaicing and denois-

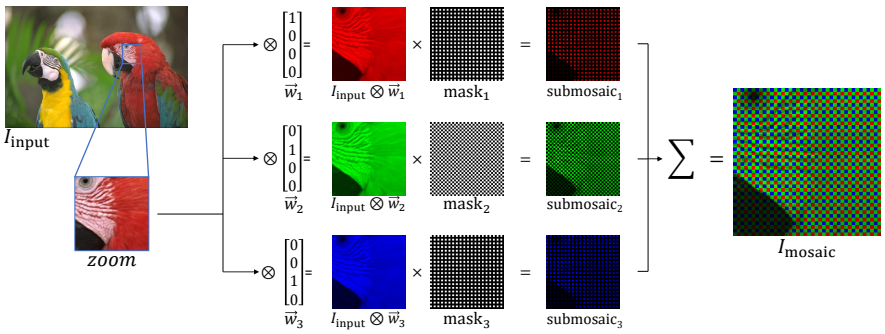


Figure 3: *Example of an encoder mimicking the Bayer pattern. Each color filter is represented by weights $\vec{w}_i = [w_{ir}, w_{ig}, w_{ib}, w_{it}]$ and a mask ($mask_i$), generating a submosaic. For the Bayer pattern, the bias term $w_{it} = 0$ for all three color filters w_i. The $I_{input} \otimes \vec{w}_i$, $submosaic_i$, and I_{mosaic} images are colored just for illustration, as they are single-channel images.*

ing [GCPD16]. All these methods were specifically designed to reconstruct Bayer filtered images.

2.3. Convolutional Neural Networks and Autoencoders

Convolutional neural networks [LBBH98] have been extensively used for image classification [KSH12], and most recently have been applied to a large variety of visual tasks. An autoencoder is a variation of a neural network that tries to learn a representation of its input in a lower-dimensional space and then reproduce the original information from such a sparse representation. Introduced in the CNN literature as a data-driven compression method [KW13], the autoencoder concept has already been used for image denoising [VLL*10], data visualization [vdMH08], superresolution [ZYW*15], and to learn priors used for image reconstruction [CJN*17].

Residual architectures (with skip connections) have been used in several image-processing tasks such as denoising [ZZC*17], superresolution [TAG*17], and others [JMFU17]. Our architecture similarly makes use of skip connections, but instead of merging branches by summing feature maps (as in the original ResNet [HZRS16]), we do so by concatenating the skipped and original feature maps. Such a choice has already been used in recent works [SLJ*15, SVI*15].

3. Joint Design of CFAs and Demosaicing

Our joint design of CFA pattern and demosaicing is expressed as the training of an autoencoder. Given a set of training images, the encoding procedure consists of projecting the corresponding input RGB information on the (trainable) color filter array pattern (Section 3.1). Such a projection generates a single-channel multispectral image mosaic (Fig. 1 (center)), which serves as input for the decoding (color reconstruction) step.

The training process minimizes a loss function defined as the mean squared error (MSE) between the provided ground truth and the reconstructed color images. The trainable parameters are the colors of the CFA pattern (encoder) and the CNN weights for demosaicing (decoder). Fig. 1 illustrates the concept. When designing the encoder, we are restricted by physical limitations imposed by the construction of actual CFAs, which precludes the use of non-linear activation functions and stacked layers. The decoder, however, may use as many layers as desired, as it is computed after the sampling process. The architecture of our network is detailed in Section 3.1.

Using only convolutional, ReLU and batch-normalization layers [IS15], our CNN architecture supports images of different sizes. By avoiding the use of fully-connected layers, the network can be trained using small image patches (with 128×128 pixels) and still reconstruct images of variable resolution (without the need of breaking the image into patches). This provides our CNN great flexibility and significantly reduces training time.

3.1. Our Autoencoder Architecture

Encoding: Our architecture optimizes colors over the entire RGB color-space. Each component (*i.e.*, color filter) of the CFA pattern is represented by a four-dimensional vector $\vec{w} = [w_r, w_g, w_b, w_t]$. The weights $[w_r, w_g, w_b] \in \mathbb{R}^3_{\geq 0}$ are RGB coefficients that represent the actual color filter (Fig. 2), and $w_t \in \mathbb{R}$ is a bias term. Appendix A shows the full vectors $\vec{w}_i$ associated with our CFAs. Given a pixel from an input image with RGB coefficients $\vec{p} = [p_r, p_g, p_b] \in \mathbb{R}^3_{\geq 0}$, we model the encoding of $\vec{p}$ by the color filter $\vec{w}$ using an affine functional $\otimes$ defined as:

$$\vec{p} \otimes \vec{w} := p_r w_r + p_g w_g + p_b w_b + w_t. \quad (1)$$

Eq. (1) is modeled as a convolution of the input image with a $1 \times 1 \times 3$ kernel plus a bias term. Note that since all coefficients of $\vec{w}$ are trainable, we must enforce non-negative weights (*i.e.*, $w_r, w_g, w_b \geq 0$) to guarantee that the color filter is physically realizable. For this, after each update, all negative weights are clamped to zero. We do not constrain the value of the bias parameter w_t since it is added after image capture. Similarly, we do not constrain the maximum value of the weights w_r, w_g, and w_b to allow for a wider range of admissible parameters during training (any constant rescaling may be performed after image capture as well). While such restrictions could be included on the architecture, having them could hamper the training convergence (by reducing the optimizer's search-space). In practice, however, the weights produced by our method seem to always fall in the $[0, 1]$ interval (see Fig. 2).

Each color filter w_i has an associated binary mask (mask_i) that specifies the pixels projected through it. Thus, simulating a CFA containing N distinct color filters requires the use of N distinct functionals and N disjoint binary masks. The projected-submosaic generated by the i-th color filter is then defined as

$$\text{submosaic}_i = \left(I_{\text{input}} \otimes \vec{w}_i\right) \times \text{mask}_i, \tag{2}$$

where I_{input} is an RGB input image, $\vec{w}_i = [w_{ir}, w_{ig}, w_{ib}, w_{it}]$, mask_i is the binary mask corresponding to the i-th CFA color, and both the functional $\otimes$ and product $\times$ are evaluated pixelwise. Thus, the multispectral mosaic generated by a CFA with N color filters is define as

$$I_{\text{mosaic}} = \sum_{i=1}^{N} \text{submosaic}_i. \tag{3}$$

Note that one can define CFAs of arbitrary sizes while enforcing how the pattern should repeat by using the disjoint masks. During training, the weights $\vec{w}_i$ are optimized to minimize color-reconstruction error, while all masks remain fixed. Appendix B describes the construction of the binary masks used in our experiments. Fig. 3 shows an example of an encoder mimicking the Bayer pattern, for which the bias $w_{it} = 0$ for all i. Note that the mask corresponding to the green component is twice as dense as the others.

Decoding: The decoding step receives as input the multispectral image mosaic I_{mosaic} produced by the encoder (Eq. (3)) and tries to reconstruct the original RGB image I_{input}. The decoder architecture consists of stacked convolutional layers, each one followed by a batch-normalization layer [IS15] and by the ReLU activation function [NH10]. All convolutional layers use 3×3 kernels, using padding to ensure the same xy-dimensions for all receptive fields. We trained multiple architectures using different numbers of skip connections. The results of these experiments suggest that the use of a single skip connection (from the beginning of the decoder to its end) results in better performance, both in terms of faster convergence and better reconstruction. All results presented in Section 4 and in the supplemental materials were obtained with such an architecture, which is shown in Fig. 4.

In addition to the monochromatic image mosaic, we provide the following additional inputs to the decoder: the submosaics of each color filter w_i, and linearly-interpolated versions of each submosaic. Although the submosaics themselves do not add new information to the decoding sub-network, they save the effort of learning how to separate individual channels, thus reducing training time. The linearly-interpolated versions of the submosaic, in turn, result in better results and faster convergence, since such an initial guess for the color-reconstructed image is much closer to the target image. In our CNN architecture, linear interpolation is achieved by convolving the submosaics with a tent kernel. Since this kernel is separable, the 2D interpolation kernel $k_{n,m}$ is defined as the outer product $k_n k_m^T$, where

$$k_c = \left[{}^1\!/_c \quad {}^2\!/_c \quad \ldots \quad {}^{(c-1)}\!/_c \quad 1 \quad {}^{(c-1)}\!/_c \quad \ldots \quad {}^2\!/_c \quad {}^1\!/_c \right]^T.$$

The vector $k_c \in \mathbb{R}^{2c-1}$ defines a 1D tent kernel that interpolates a mosaic generated by a 1D CFA pattern containing c colors. For example, $k_2 = [{}^1\!/_2 \; 1 \; {}^1\!/_2]$ and, for a 4×4 CFA pattern with 16 distinct color filters, the required tent kernel is

$$k_{4,4} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 3 & 4 & 3 & 2 & 1 \\ 2 & 4 & 6 & 8 & 6 & 4 & 2 \\ 3 & 6 & 9 & 12 & 9 & 6 & 3 \\ 4 & 8 & 12 & 16 & 12 & 8 & 4 \\ 3 & 6 & 9 & 12 & 9 & 6 & 3 \\ 2 & 4 & 6 & 8 & 6 & 4 & 2 \\ 1 & 2 & 3 & 4 & 3 & 2 & 1 \end{bmatrix}.$$

This interpolation is performed by a standard convolutional layer with fixed kernel weights. Although these weights could also be learned, we have found that fixing them (*i.e.*, not updating them during training) produces better results.

The complete architecture of our autoencoder is depicted in Fig. 4, while Section 4 provides implementation details. Besides the skip connection from the beginning of the decoder to its end (for improving color reconstruction and training convergence) indicated by the dotted orange arrow, we also use skip connections from each submosaic and corresponding linearly-interpolated versions to the decoder's input. Such connections are indicated by the dotted green arrows and proved to speed up the training, providing a path for gradient backpropagation.

Training: Given the number of color filters for the CFA and the topology of the decoding network (number of convolutional layers with their inner parameters – Fig. 4), the autoencoder is trained by iteratively feeding patches to the network. Such patches are used to update the the colors of the CFA and weights of the demosaicing method, trying to minimize the MSE between the input color patches and their reconstructions.

4. Demosaicing Results and Evaluation

Our particular instantiation of the autoencoder architecture described in Section 3.1 and illustrated in Fig. 4 includes a decoder consisting of a stack of 12 convolutional layers. The number of (3×3) kernels used in each of these 12 layers are [64, 64, 64, 64, 64, 64, 128, 128, 128, 128, 128, 128], respectively. This setup presents a good trade-off between network expressiveness and training time. We show that the quality of our reconstructions surpasses the ones generated by existing methods [Cha16, MBP*09, GCPD16].

We implemented our network using Keras [Cho15], running on top of Theano [Tea16], using MSE as the loss function, and the Adam optimizer [KB14] ($lr = \alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$), with batch size of 32. Training was performed using two datasets provided by Gharbi et al. [GCPD16]: *vdp* and *moiré*. We used the same images as Gharbi et al. for training, consisting of 2,590,186 128×128-pixel patches. In addition, we used horizontal and vertical flips, as well as random 90° rotations, for data augmentation. Our model for reconstructing noise-free images (as well as our demosaicing method for the Bayer pattern) was trained for 3 epochs, which corresponds to approximately 5 days of training time on a GeForce GTX TITAN X GPU. Our model for reconstructing noisy data was trained for 6 epochs.

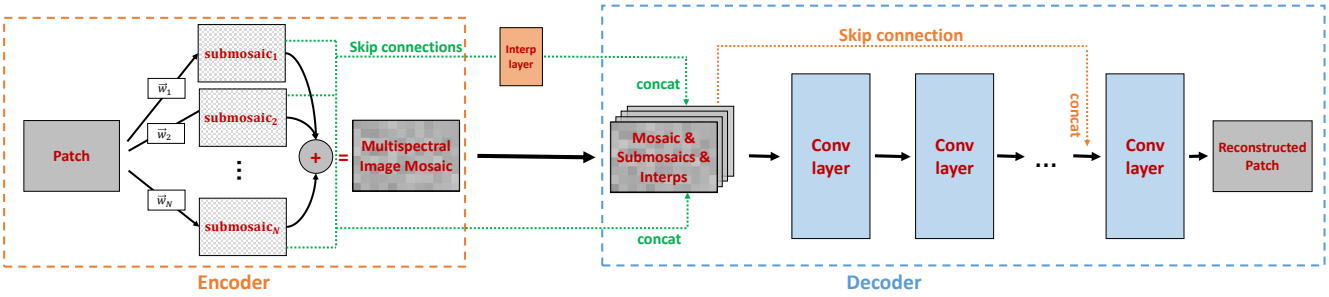


Figure 4: *Our autoencoder architecture. The encoding step projects the colored image patches through the trainable CFA, generating a multispectral image mosaic. Skip connections (green arrows) contribute submosaics consisting of the separate color channels as well as per-channel interpolated images, which are stacked with the multispectral image mosaic forming a deeper representation (total of 2N + 1 channels for N color filters).The decoding component is based on residual blocks. This autoencoder can produce CFA patterns of different sizes and works with networks of distinct depths. The beginning-to-end decoder skip connection (orange arrow) improves color reconstruction and training convergence. Parameters details are given in Section 3.1.*

Demosaic (Bayer CFA)	Kodak	McMaster	vdp	moiré
Bilinear	29.51	32.32	24.97	27.39
[ZWBL11a]	35.66	29.87	24.85	28.04
[Get11a]]	35.98	35.87	29.88	31.70
[BCMS09]	36.62	35.24	29.34	31.30
[LKV10]	37.17	32.22	27.67	28.70
[CM12]	38.51	33.29	29.03	30.96
[JAJ*14]	38.71	36.84	30.27	31.75
[HST*14]	38.83	38.30	30.93	34.61
[KMTO16]	38.84	36.86	30.52	31.90
[JD13]	40.03	33.78	29.34	31.33
[Get11b]	40.13	34.17	30.06	32.25
[MBP*09]	41.23	36.13	30.94	33.16
[GCPD16]	41.79	39.14	33.96	36.64
Our 2x2 Bayer	41.86	39.51	34.28	36.33
Demosaic (non-Bayer CFA)	**Kodak**	**McMaster**	**vdp**	**moiré**
[Cha16]	31.52	28.05	23.97	23.97
[CFZ14]	33.51	30.94	25.91	28.77
[Con11]	38.10	32.90	28.65	30.45
[HLLD11]	39.42†	—	—	—
[BLLY16]	40.24†	—	—	—
[HW08]	40.36†	—	—	—
[LBLY17]	41.59†	—	—	—
Our 4x4 noise-free	**43.13**	**40.18**	**35.17**	**37.70**

Table 1: *Comparison of our* 4 × 4 *noise-free CFA and demosaicing technique against existing methods. The numbers show the average PSNR values of reconstructions for four datasets. All results were generated using code provided by the authors, except the ones marked with* †*, whose numbers were taken from the corresponding publications. Our* 4 × 4 *noise-free CFA and demosaicing outperform all other techniques in all four datasets (best results in bold). Our demosaicing network for the Bayer pattern also outperforms all previous techniques for the Kodak, McMaster, and vdp datasets, and got very close to Gharbi et al.'s in the moiré dataset.*

4.1. Comparisons to Other Approaches

Table 1 compares the PSNR of the results obtained with our 4 × 4 noise-free CFA (Fig. 2) with the ones produced by the most successful demosaicing techniques [MBP*09, Get11b, BLLY16, HLLD11,Con11,Wan14,Cha16,GCPD16]. For all comparisons, we have used either source or executable code provided by the authors. All images were saved to disk to guarantee similar color quantization and later compared based on PSNR (error averaged over pixels and color channels before computing the logarithm). In addition, all measurements were performed on full-resolution images (borders included). We did not use any of the test images in our training phase. Table 1 shows the average PSNR for the traditional Kodak [Fra99] and McMaster [ZWBL11b] datasets, as well as for the two datasets of Gharbi et al. (*vdp* and *moiré*) [GCPD16]. The techniques on the top portion of Table 1 perform demosaicing for the Bayer pattern, while the ones on the bottom portion perform demosaicing for non-Bayer CFAs. Our 4 × 4 noise-free CFA and demosaicing solution (last row of Table 1) surpasses all existing methods in all four datasets (even for Kodak and McMaster, from which no images were used for training). We also trained our architecture using the Bayer pattern, *i.e.*, only optimizing the decoder (Fig. 3). Our demosaicing network for the Bayer pattern also outperforms all previous techniques for the Kodak, McMaster, and *vdp* datasets, and got very close to Gharbi et al.'s [GCPD16] in the *moiré* dataset. These results clearly demonstrate the effectiveness of our autoencoder architecture and the advantage of jointly optimizing CFA design and demosaicing.

Fig. 5 compares the reconstruction quality of our 4 × 4 noise-free model with the state-of-the-art demosaicing techniques [Cha16, MBP*09, HST*14, GCPD16]. For such comparison, we have used code provided by the authors for their noise-free trained models. Chakrabarti's method [Cha16] does not reconstruct the full patch, so we measure the PSNR only for the reconstructed area. The examples in Fig. 5 are from Gharbi et al.'s datasets. Note how our method better handles high-frequency information, being less susceptible to aliasing artifacts than other techniques (see the stripped shirt example in Fig. 5). Additional examples can be found in the supplementary materials.

Chakrabarti [Cha16] trained and tested his model using the dataset of Shi and Funt [SF10]. We also tested our 4 × 4 noise-free CFA on the same test images. Chakrabarti [Cha16] achieves an average PSNR of 41.50, while our model achieves 48.96, a signifi-

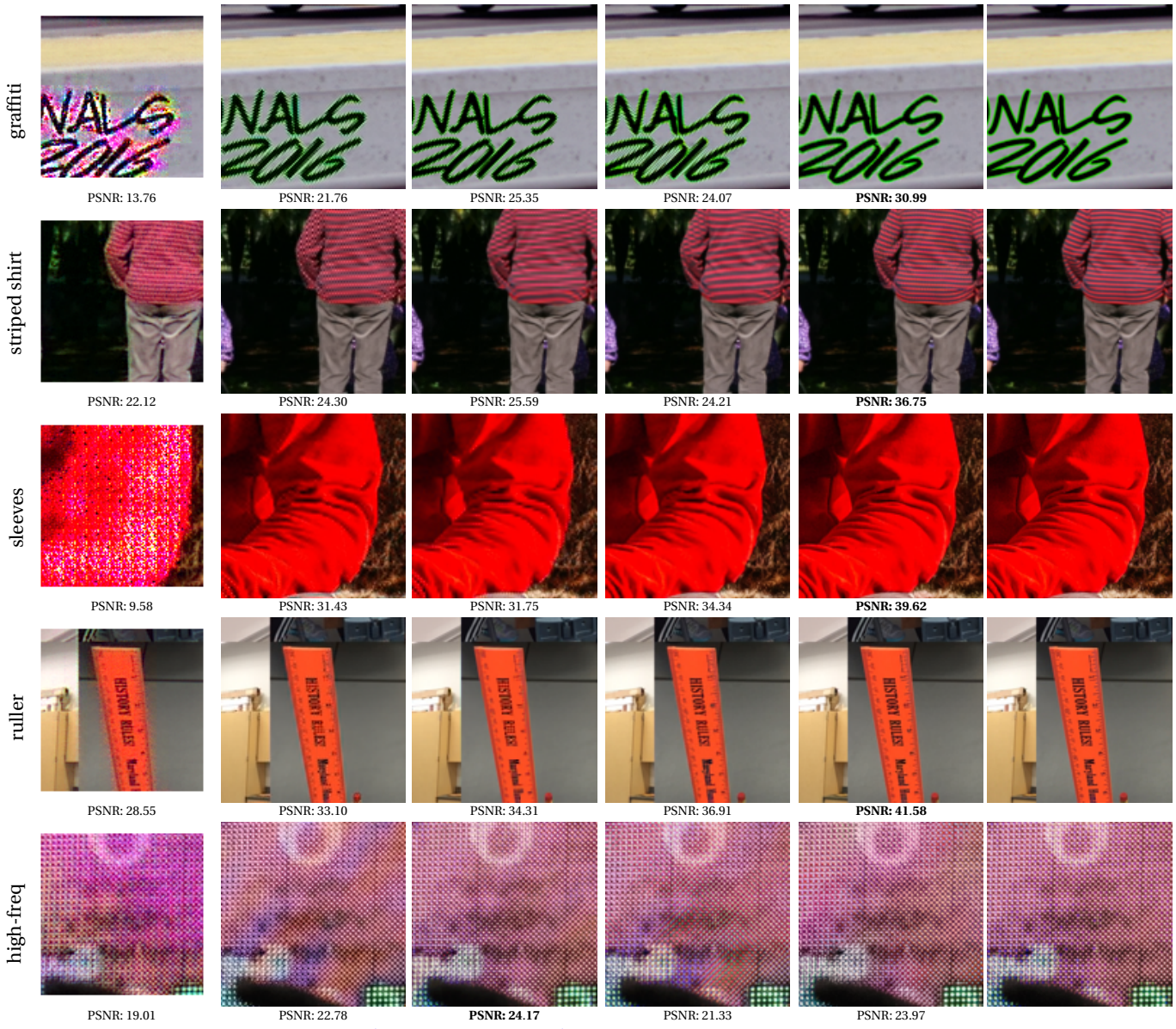


Figure 5: *Comparison of reconstruction quality of our* 4×4 *noise-free method and the state-of-the-art techniques (best PSNR values shown in bold). Patches from Gharbi et al.'s datasets [GCPD16]. Better visualized in the digital version.*

cant improvement in reconstruction quality, even though no images from Shi and Funt's dataset were used for training our CFA.

Table 2 shows the average running times of the state-of-the-art techniques for reconstructing images from the Kodak dataset (resolution of 768×512). All measurements were made on an Intel Core i7-2660 CPU and GeForce GTX TITAN X GPU. Our technique is slightly slower than other GPU-based implementations, but still much faster than methods based on compressive sensing [MBP*09] or optimization schemes [HST*14].

Running Times (seconds)	
Chakrabarti (GPU) [Cha16]	0.09
Gharbi et al. (GPU) [GCPD16]	0.16
Our 4x4 (GPU)	0.34
Mairal et al. (CPU) [MBP*09]	565.23
Heide et al. (CPU) [HST*14]	652.63

Table 2: *Average running times of state-of-the-art techniques for reconstructing images from the Kodak dataset.*

4.2. Reconstruction in the presence of noise

The idea of jointly performing demosaicing and denoising has been explored by many works [Con11, CM12, HvLP12, KHKP16, GCPD16, Cha16]. Enhancing our autoencoder with denoising capabilities only requires feeding the network with patches corrupted by (artificial) noise during the training phase, while comparing the network's output to the noise-free images. To demonstrate the flexibility of our architecture, we have trained the same network structure from scratch, using the same datasets used for training our 4×4 noise-free CFA. This time, however, each input patch was corrupted by additive Gaussian noise. Although in linear space camera noise should be modeled as a combination of Poissonian and Gaussian noise [FTKE08], according to Jeon and Dubois [JD13], for white-balanced, gamma-corrected images (such as the case of the vdp and moiré datasets [GCPD16] used for training) one can model noise as signal-independent white Gaussian noise. By corrupting the images with Gaussian noise with a varying standard deviation randomly picked from the set $\{0, 4, 8, 12, 16, 20\}$, we avoid the need of specialized networks for each noise level (such as in Chakrabarti [Cha16]), training a single model that handles a large range of noise variance.

Table 3 compares the PSNR for our 4×4 CFA for noisy data (Fig. 2 (right)) against existing techniques. The quality of our reconstructions surpasseses previous approaches in all datasets, for all noise levels. Moreover, unlike Gharbi et al.'s approach [GCPD16], ours does not require an estimate of the noise level, and thus the quality of our denoising results are not dependent on the accuracy of any noise estimation step.

Fig. 6 compares our results to the the state-of-the-art techniques. Note that our method performs an optimization that jointly improves CFA design, demosaicing, and denoising. As a result, it can reduce noise without oversmoothing the images, generating higher-quality reconstructions. Our CFA pattern for noisy datasets can be seen in Fig. 2 (right). Additional examples can be found in the supplementary materials.

Noise $\sigma = 4$	**Kodak**	**McMaster**	**vdp**	**moiré**
[Cha16]	28.59	26.32	21.96	21.72
[CFZ14]	30.70	28.34	25.34	27.71
[Con11]	34.15	31.19	27.86	29.30
[CM12]	34.43	31.53	28.19	29.69
[GCPD16]	36.90	36.02	31.61	33.31
Our 4x4 noise	**38.01**	**36.59**	**32.83**	**34.54**
Noise $\sigma = 8$	**Kodak**	**McMaster**	**vdp**	**moiré**
[Cha16]	26.63	25.16	20.79	20.52
[CFZ14]	27.26	25.56	23.75	25.39
[Con11]	29.83	28.50	26.26	27.20
[CM12]	30.14	28.84	26.56	27.52
[GCPD16]	34.19	33.97	29.87	31.34
Our 4x4 noise	**35.08**	**34.39**	**31.16**	**32.50**
Noise $\sigma = 12$	**Kodak**	**McMaster**	**vdp**	**moiré**
[Cha16]	25.59	24.32	20.22	20.04
[CFZ14]	24.62	23.41	22.20	23.34
[Con11]	26.74	26.16	24.59	25.19
[CM12]	27.07	26.50	24.87	25.49
[GCPD16]	32.40	32.41	28.39	29.87
Our 4x4 noise	**33.31**	**32.90**	**29.73**	**31.02**
Noise $\sigma = 16$	**Kodak**	**McMaster**	**vdp**	**moiré**
[Cha16]	25.28	23.96	20.16	20.26
[CFZ14]	22.60	21.68	20.81	21.65
[Con11]	24.44	24.23	23.06	23.45
[CM12]	24.76	24.56	23.32	23.74
[GCPD16]	31.07	31.19	27.18	28.73
Our 4x4 noise	**32.17**	**31.81**	**28.56**	**29.88**
Noise $\sigma = 20$	**Kodak**	**McMaster**	**vdp**	**moiré**
[Cha16]	24.60	23.40	19.98	20.31
[CFZ14]	20.93	20.24	19.60	20.23
[Con11]	22.61	22.62	21.70	21.96
[CM12]	22.93	22.94	21.95	22.23
[GCPD16]	30.00	30.15	26.17	27.80
Our 4x4 noise	**31.20**	**30.87**	**27.57**	**28.93**

Table 3: *Comparison of our model with existing methods for joint denoise and demosaic. The numbers show the average PSNR values of reconstructions for four datasets corrupted by noise of different intensities. Our model outperforms all other techniques in all four datasets and for all noise intensities.*

5. Discussion

We evaluated several network architectures beyond the ones presented, and some observations worthy mentioning:

Deepness versus wideness: We tested shallower and deeper, as well as thinner and wider networks. Contrary to many works that claim that deepness is the key for good performance, we have found similar results to Zagoruyko and Komodakis [ZK16], suggesting that wideness is as important as deepness.

Skip connections: We have used skip connections to stack submosaic images for the decoder, which improved the performance and convergence of the network. We have also tested different architectures using distinct dispositions, observing no correlation between the number of connections and higher PSNR.

Additional input: By providing additional information to the demosaicing (decoder) scheme, our network is capable of learning faster and achieving better reconstructions. In addition to the mosaic image, we have also provided the CFA submosaics, and their linearly interpolated versions. This simplifies the training, as the network does not need to learn to separate each color submosaic, nor the interpolation kernels from scratch.

Masks: We use disjoint binary masks to enforce pattern repetition, and to be able to handle images of arbitrary sizes. But the masks can be used to impose additional constraints. For instance, one can use them to find 2×2 patterns with only three colors (as in the Bayer pattern), or to enforce designs that follow specific patterns, such as blue-noise characteristics [Con10], or diagonal designs [LP05, BLLY16].

CFA pattern size: Our architecture can train CFAs with an arbitrary number of color filters, and we have opted to train designs larger than the traditional 2×2 patterns. Training bigger-sized CFA patterns has several advantages. First, they contain a larger number of distinct colors and thus provide more coverage dur-

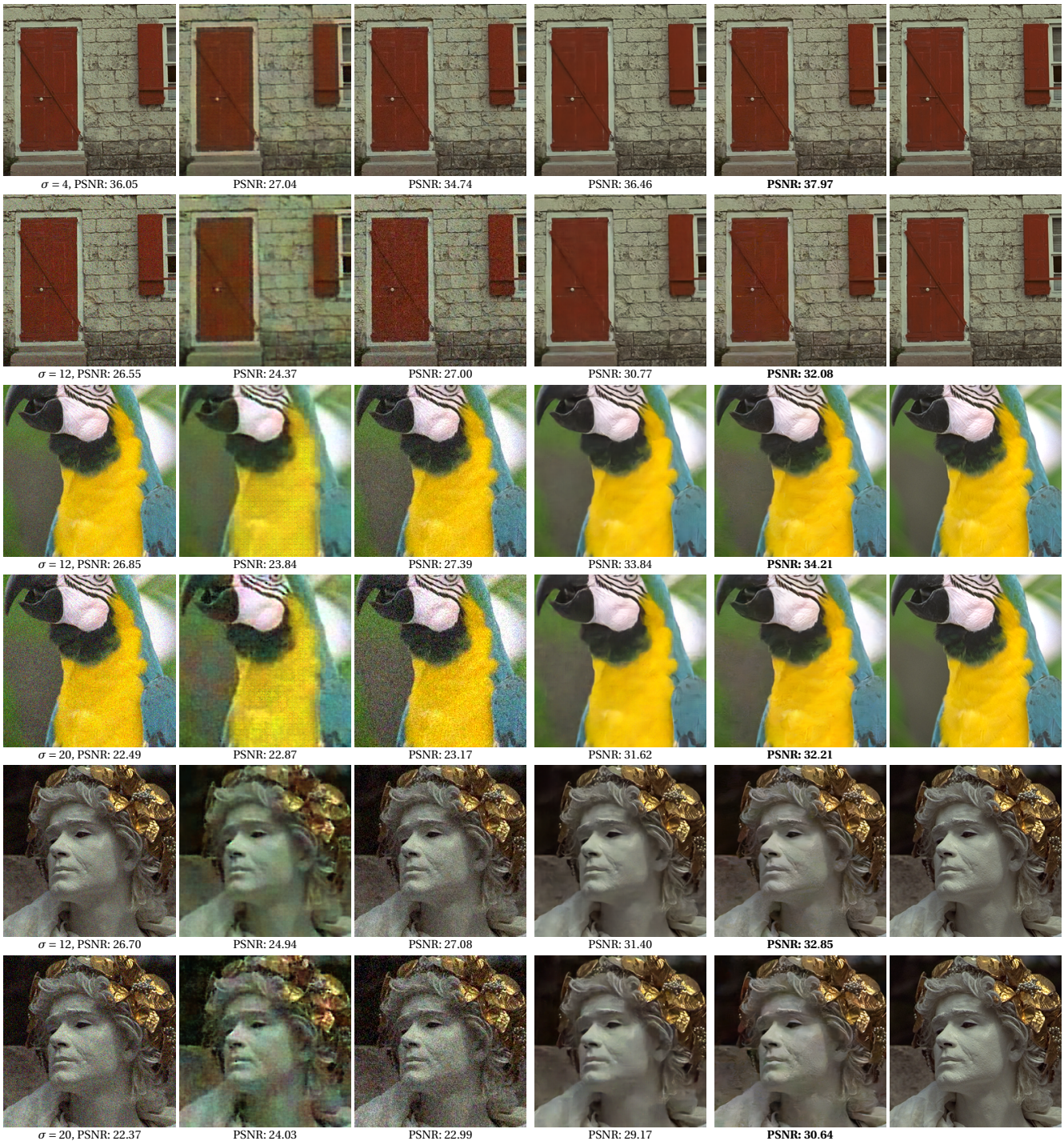


Figure 6: *Comparison of the reconstructions obtained by our method and state-of-the-art techniques that jointly perform denoising and demosaicing (best PSNR in bold). The input images were corrupted with Gaussian noise (left column), whose level is indicated by the standard deviation σ (in RGB* [0,255] *units). Images from the Kodak dataset. Better visualized in the digital version.*

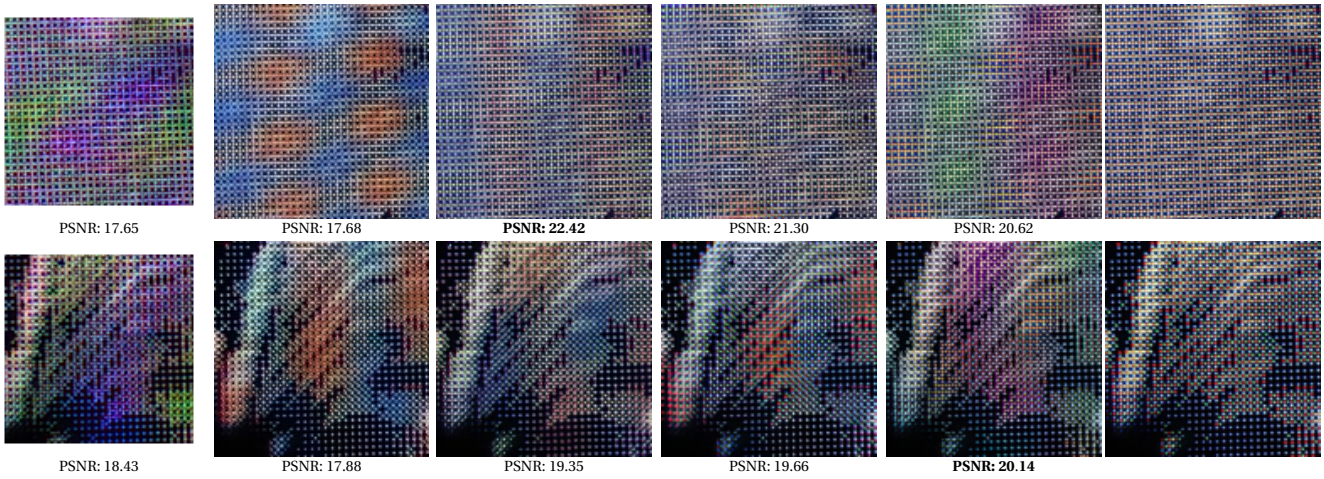


Figure 7: *Images with extreme high-frequencies are challenging for all demosaicing methods, whose results exhibit aliasing artifacts. These patches are from the moiré dataset [GCPD16].*

ing sampling of the color space, allowing the CNN to make better use of correlations among colors. Second, smaller CFAs are more susceptible to aliasing due to pattern repetition, while bigger CFAs allow the learning of more stochastic patterns. Third, from an optimization perspective, the 2×2 search-space is a subspace of the 4×4 search-space, meaning that a 4×4 CFA can learn a 2×2 pattern if it is advantageous. For instance, a careful inspection of Fig. 2 reveals that each of our 4×4 CFAs actually consists of two side-by-side copies of a 4×2 pattern. The small differences among the corresponding RGB coefficients in the 4×2 patterns in each CFA are fairly small, and are likely to be reduced with longer training. This suggests that 4×2 patterns are the most efficient tileable representations for CFAs achievable with a 4×4 pattern.

Manufacturing our CFAs: Our encoder optimizes the CFA colors over the full RGB space. Although our 4×4 CFAs do not use the standard Bayer color filters, the colors used in our patterns are a linear combination of these standard filters, which should simplify the manufacturing process. Alternatively, Chiulli [Chi89] has patented a technique for creating color filters from any combinations of red, green, blue, cyan, yellow and magenta dies. More recently, SILIOS Technologies has developed a manufacturing technique called COLOR SHADES® for producing band-pass filters [SIL17]. This technology combines thin film deposition and micro/nano-etching processes onto a silica substrate [LWTG14]. COLOR SHADES® provides band-pass filters in the visible range from 400 nm to 700 nm (as well as in the IR range). Lapray et al. [LWTG14] describe the construction of a multispectral CFA using eight optical filter bands produced with COLOR SHADES®, and compare the simulated and measured responses of the individual filters. This technology could be used to produce our CFAs.

Demosaicing of extremely high-frequency content is challenging to all demosaicing methods. Fig. 7 shows examples of two image patches the for which all techniques, including ours, are unable to obtain high-quality image reconstructions. Such problems are due to aliasing, when color high-frequency details cannot be appropriately sampled by the CFA [GCPD16]. Note that the artifacts in the reconstructions by the techniques of Mairal et al. and Gharbi et al. are similar. Both use the same Bayer CFA, indicating that such moiré artifacts are due to CFA color subsampling, rather than to the demosaicing method itself.

Before arriving at the described architecture, we have systematically tried many alternatives. Such exploration included the use of L1 and L2 regularizers, different optimizers (Adadelta and Adam), dropouts [SHK*14], various configurations of skip-connections, different sizes of CFA patterns (including 6×6 and 8×8), and trainable/fixed interpolation layers. While testing all combinations of these elements is unfeasible, we have made extensive experimentation. The results of these tests indicated that the architecture for the 4×4 patterns (both for noise-free and noisy patterns) achieved the overall best PSNR results. Note that the CFA designs learned for the noise-free and for the noisy cases are similar, one being a shifted version of the other. This indicates that those colors were not found by chance, and they indeed provide lower reconstruction errors.

6. Conclusion

We have presented a convolutional neural network architecture for performing joint design of color filter arrays, demosaicing, and denoising. By expressing the CFA projection and linear interpolation as convolutional layers, our network finds the filter pattern and corresponding demosaicing method that jointly minimize image reconstruction error. The patterns and algorithms produced by our method provide high-quality color reconstructions, surpassing the state-of-the-art techniques on all standard demosaicing datasets.

Our approach can also reconstruct high-quality images from noisy data, outperforming existing techniques for all noise levels, without requiring any information about the noise level in the input data. In addition, it can be used to obtain effective demosaicing strategies for existing CFA patterns. Given its flexibil-

ity, our architecture might possibly be adapted for the design of CFAs for capturing HDR content with a single shot.

Acknowledgements

We would like to thank the reviewers for their insightful comments, and NVIDIA for donating the GeForce GTX Titan X GPU used for this research. This work was sponsored by CNPq-Brazil (fellowships and grants 306196/2014-0, 423673/2016-5).

References

[ASH05] ALLEYSSON D., SUSSTRUNK S., HERAULT J.: Linear demosaicing inspired by the human visual system. *IEEE TIP 14*, 4 (2005), 439–449. 1, 2

[Bay76] BAYER B.: Color imaging array, 1976. US Patent 3,971,065. URL: https://www.google.com/patents/US3971065. 1, 2

[BCMS09] BUADES A., COLL B., MOREL J. M., SBERT C.: Self-similarity driven color demosaicking. *IEEE TIP 18*, 6 (2009), 1192–1202. 2, 5

[BLLY16] BAI C., LI J., LIN Z., YU J.: Automatic design of color filter arrays in the frequency domain. *IEEE TIP 25*, 4 (2016), 1793–1807. 1, 2, 5, 7

[CFZ14] CHAKRABARTI A., FREEMAN W. T., ZICKLER T.: Rethinking color cameras. In *ICCP* (2014), pp. 1–8. 2, 5, 7

[Cha16] CHAKRABARTI A.: Learning sensor multiplexing design through back-propagation. *CoRR abs/1605.07078* (2016). 2, 4, 5, 6, 7, 8, 9

[Chi89] CHIULLI C.: Method for manufacturing an optical filter, Feb. 28 1989. US Patent 4,808,501. 9

[Cho15] CHOLLET F.: Keras: Deep learning library for theano and tensorflow, 2015. https://keras.io/. 4

[CJN*17] CHOI I., JEON D. S., NAM G., GUTIERREZ D., KIM M. H.: High-quality hyperspectral reconstruction using a spectral prior. *ACM TOG 36*, 6 (Nov. 2017), 218:1–218:13. 3

[CM12] CONDAT L., MOSADDEGH S.: Joint demosaicking and denoising by total variation minimization. In *IEEE ICIP* (2012), pp. 2781–2784. 2, 5, 7, 8

[Con09] CONDAT L.: A new random color filter array with good spectral properties. In *ICIP* (2009), pp. 1613–1616. 2

[Con10] CONDAT L.: Color filter array design using random patterns with blue noise chromatic spectra. *Image and Vision Computing 28*, 8 (2010), 1196 – 1202. 2, 7

[Con11] CONDAT L.: A new color filter array with optimal properties for noiseless and noisy color image acquisition. *IEEE TIP 20*, 8 (2011), 2200–2210. 1, 2, 5, 7

[Dub05] DUBOIS E.: Frequency-domain methods for demosaicking of Bayer sampled color images. *IEEE Signal Processing Lett* (2005), 847–850. 1, 2

[DVM16] DAVE A., VADATHYA A. K., MITRA K.: Compressive image recovery using recurrent generative model. *CoRR abs/1612.04229* (2016). URL: http://arxiv.org/abs/1612.04229. 2

[Fra99] FRANZEN R.: Kodak lossless true color image suite, 1999. http://r0k.us/graphics/kodak/. 5

[FTKE08] FOI A., TRIMECHE M., KATKOVNIK V., EGIAZARIAN K.: Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. *IEEE TIP 17*, 10 (2008), 1737–1754. 7

[GCPD16] GHARBI M., CHAURASIA G., PARIS S., DURAND F.: Deep joint demosaicking and denoising. *ACM ToG 35*, 6 (2016), 191:1–191:12. 3, 4, 5, 6, 7, 8, 9

[Get11a] GETREUER P.: Color demosaicing with contour stencils. In *ICDSP* (2011), pp. 1–6. 5

[Get11b] GETREUER P.: Zhang-Wu Directional LMMSE Image Demosaicking. *Image Processing On Line 1* (2011). 1, 5

[GSL00] GO J., SOHN K., LEE C.: Interpolation using neural networks for digital still cameras. *IEEE TCE 46*, 3 (2000), 610–616. 2

[HLLD11] HAO P., LI Y., LIN Z., DUBOIS E.: A geometric method for optimal design of color filter arrays. *IEEE TIP 20*, 3 (2011), 709–722. 1, 2, 5

[HST*14] HEIDE F., STEINBERGER M., TSAI Y.-T., ROUF M., PAJAK D., REDDY D., GALLO O., LIU J., HEIDRICH W., EGIAZARIAN K., KAUTZ J., PULLI K.: Flexisp: A flexible camera image processing framework. *ACM ToG 33*, 6 (2014). 2, 5, 6, 9

[HvLP12] HEINZE T., VON LÖWIS M., POLZE A.: Joint multi-frame demosaicing and super-resolution with artificial neural networks. In *IWSSIP* (2012), pp. 540–543. 1, 2, 7

[HW08] HIRAKAWA K., WOLFE P. J.: Spatio-spectral color filter array design for optimal image recovery. *IEEE TIP 17*, 10 (2008), 1876–1890. 2, 5

[HZRS16] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. In *CVPR* (2016). 3

[IS15] IOFFE S., SZEGEDY C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML* (2015), pp. 448–456. 3, 4

[JAJ*14] JAISWAL S. P., AU O. C., JAKHETIYA V., YUAN Y., YANG H.: Exploitation of inter-color correlation for color image demosaicking. In *IEEE ICIP* (2014), pp. 1812–1816. 2, 5

[JD13] JEON G., DUBOIS E.: Demosaicking of noisy bayer-sampled color images with least-squares luma-chroma demultiplexing and noise level estimation. *IEEE TIP 22*, 1 (2013), 146–156. 5, 7

[JMFU17] JIN K. H., MCCANN M. T., FROUSTEY E., UNSER M.: Deep convolutional neural network for inverse problems in imaging. *IEEE TIP 26*, 9 (2017), 4509–4522. 3

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014). 4

[KB15] KAUR S., BANGA V. K.: A survey of demosaicing: Issues and challenges. *International Journal of Science, Engineering and Technologies 2*, 1 (2015). 2

[KHKP16] KLATZER T., HAMMERNIK K., KNOBELREITER P., POCK T.: Learning joint demosaicing and denoising based on sequential energy minimization. In *IEEE ICCP* (2016), pp. 1–11. 2, 7

[KHO00] KAPAH O., HEL-OR H. Z.: Demosaicking using artificial neural networks. In *Proc. SPIE* (2000), vol. 3962, pp. 112–120. 2

[KMTO16] KIKU D., MONNO Y., TANAKA M., OKUTOMI M.: Beyond color difference: Residual interpolation for color image demosaicking. *IEEE TIP 25*, 3 (2016), 1288–1300. 5

[KSH12] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: Imagenet classification with deep convolutional neural networks. In *NIPS 25*. 2012, pp. 1097–1105. 3

[KW13] KINGMA D. P., WELLING M.: Auto-encoding variational bayes. *CoRR abs/1312.6114* (2013). 2, 3

[LBBH98] LECUN Y., BOTTOU L., BENGIO Y., HAFFNER P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324. 3

[LBLY17] LI J., BAI C., LIN Z., YU J.: Optimized color filter arrays for sparse representation-based demosaicking. *IEE TIP 26*, 5 (2017), 2381–2393. 1, 2, 5

[LCTZ07] LIAN N. X., CHANG L., TAN Y. P., ZAGORODNOV V.: Adaptive filtering for color filter array demosaicking. *IEEE TIP 16*, 10 (2007), 2515–2525. 2

[LGZ08] LI X., GUNTURK B., ZHANG L.: Image demosaicing: a systematic survey. In *Proc. SPIE* (2008), vol. 6822, pp. 1–15. 2

[LH06] LONG Y., HUANG Y.: Adaptive demosaicking using multiple neural networks. In *IEEE Signal Processing Society Workshop on MLSP* (2006), pp. 353–357. 2

[Li05] LI X.: Demosaicing by successive approximation. *IEEE TIP 14*, 3 (2005), 370–379. 2

[LKV10] LU Y., KARZAND M., VETTERLI M.: Demosaicking by alternating projections: Theory and fast one-step implementation. *IEEE TIP 19*, 8 (2010), 2085–2098. 5

[LP05] LUKAC R., PLATANIOTIS K. N.: Color filter arrays: design and performance analysis. *IEEE TCE 51*, 4 (2005), 1260–1267. 2, 7

[LV09] LU Y. M., VETTERLI M.: Optimal color filter array design: quantitative conditions and an efficient search procedure. In *Digital Photography* (2009), vol. 7250, SPIE, pp. 1–9. 2

[LWTG14] LAPRAY P.-J., WANG X., THOMAS J.-B., GOUTON P.: Multispectral filter arrays: Recent advances and practical implementation. *Sensors 14*, 11 (2014), 21626–21659. 9

[MAKR13] MOGHADAM A. A., AGHAGOLZADEH M., KUMAR M., RADHA H.: Compressive framework for demosaicing of natural images. *IEEE TIP 22*, 6 (2013), 2356–2371. 2

[MBP*09] MAIRAL J., BACH F., PONCE J., SAPIRO G., ZISSERMAN A.: Non-local sparse models for image restoration. In *ICCV* (2009), pp. 2272–2279. 1, 2, 4, 5, 6, 9

[MC11] MENON D., CALVAGNO G.: Color image demosaicking: An overview. *Image Commun. 26*, 8-9 (2011), 518–533. 2

[NH10] NAIR V., HINTON G. E.: Rectified linear units improve restricted boltzmann machines. In *ICML* (2010), pp. 807–814. 4

[SF10] SHI L., FUNT B.: Re-processed version of the gehler color constancy dataset of 568 images, 2010. http://www.cs.sfu.ca/~colour/data/shi_gehler/. 5

[SHK*14] SRIVASTAVA N., HINTON G., KRIZHEVSKY A., SUTSKEVER I., SALAKHUTDINOV R.: Dropout: A simple way to prevent neural networks from overfitting. *JMLR 15*, 1 (2014), 1929–1958. 9

[SIL17] SILIOS TECHNOLOGIES: COLOR SHADES, 2017. URL: https://www.silios.com/color-shades. 9

[SLJ*15] SZEGEDY C., LIU W., JIA Y., SERMANET P., REED S., ANGUELOV D., ERHAN D., VANHOUCKE V., RABINOVICH A.: Going deeper with convolutions. In *CVPR* (2015), pp. 1–9. 3

[SVI*15] SZEGEDY C., VANHOUCKE V., IOFFE S., SHLENS J., WOJNA Z.: Rethinking the inception architecture for computer vision. *CoRR abs/1512.00567* (2015). 3

[TAG*17] TIMOFTE R., AGUSTSSON E., GOOL L. V., YANG M. H., ZHANG L., LIM B., ... GUO Q.: Ntire 2017 challenge on single image super-resolution: Methods and results. In *IEEE CVPRW* (2017), pp. 1110–1121. 3

[Tea16] TEAM T. D.: Theano: A Python framework for fast computation of mathematical expressions. *CoRR abs/1605.02688* (2016). 4

[vdMH08] VAN DER MAATEN L., HINTON G. E.: Visualizing high-dimensional data using t-sne. *JMLR 9* (2008), 2579–2605. 3

[VLL*10] VINCENT P., LAROCHELLE H., LAJOIE I., BENGIO Y., MANZAGOL P.-A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR 11* (2010), 3371–3408. 3

[Wan14] WANG Y. Q.: A multilayer neural network for image demosaicking. In *ICIP* (2014), pp. 1852–1856. 1, 2, 5

[ZK16] ZAGORUYKO S., KOMODAKIS N.: Wide residual networks. *CoRR abs/1605.07146* (2016). 7

[ZW05] ZHANG L., WU X.: Color demosaicking via directional linear minimum mean square-error estimation. *IEEE TIP 14*, 12 (2005), 2167–2178. 2

[ZWBL11a] ZHANG L., WU X., BUADES A., LI X.: Color demosaicking by local directional interpolation and nonlocal adaptive thresholding. *J. Electronic Imaging 20*, 2 (2011), 023016. 2, 5

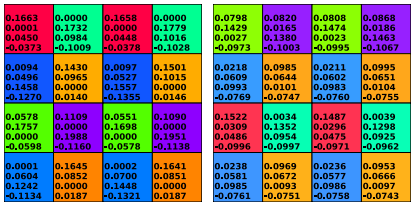
[ZWBL11b] ZHANG L., WU X., BUADES A., LI X.: Mcmaster dataset, 2011. http://www4.comp.polyu.edu.hk/~cslzhang/CDM_Dataset.htm. 5

[ZYW*15] ZENG K., YU J., WANG R., LI C., TAO D.: Coupled deep autoencoder for single image super-resolution. *IEEE Transactions on Cybernetics PP*, 99 (2015), 1–11. 3

[ZZC*17] ZHANG K., ZUO W., CHEN Y., MENG D., ZHANG L.: Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE TIP 26*, 7 (2017), 3142–3155. 3

Appendix A: RGB Weights and Bias Terms

Fig. 8 shows the RGB and bias weights ($\vec{w}_i = [w_{ir}, w_{ig}, w_{ib}, w_{it}]$) for our 4×4 CFAs for noise-free and noisy data. The values of w_{ir}, w_{ig}, w_{ib}, and w_{it} are shown from top to bottom inside each color filter cell.



Our 4x4 noise-free Our 4x4 noise

Figure 8: *Our* 4×4 *patterns for noise-free and for noisy data. The color filters are expressed using four coefficients: R, G, B and a bias term, respectively (shown inside each cell).*

Appendix B: Construction of Binary Masks

A CFA is a periodic structure, with the CFA pattern corresponding to one period. Given an $M \times N$ CFA pattern, this will result in MN (M times N) distinct $M \times N$ binary masks for the CFA pattern (*i.e.*, one binary mask for each CFA element). Each such binary mask has a single non-zero element (with value 1), at the position corresponding to the given CFA element. Thus, in a $M \times N$ CFA, the CFA element at position (i, j), $1 \leq i \leq M$, $1 \leq j \leq N$, has a corresponding binary mask containing zeros everywhere, except at mask position (i, j), which contains the value 1. Similar to a complete CFA, the actual binary masks cover the entire image, being obtained by tiling the corresponding CFA-element binary masks.