Learning Optimum Binary Color Filter Arrays for Demosaicing with Neural Networks

Cemre Omer Ayna[a] and Ali Cafer Gurbuz[a]

[a]Mississippi State University, Mississippi State, MS

ABSTRACT

Modern digital color cameras depend on Color Filter Arrays (CFA) for capturing color information. The majority of the commercial CFAs are designed by hand with different physical and application-specific considerations. The available machine learning (ML)-based CFA learning architectures dismiss the considerations of a physical camera device. This study aims to develop an alternative approach to jointly learn binary Color Filter Arrays (CFA) in a deep learning-based filtering-demosaicing pipeline. The proposed approach provides higher reconstruction performance than the compared hand-designed filters while learning physically applicable CFAs. This paper includes the learned binary CFAs for various color configurations and training data size, their analysis with common reconstruction metrics, and a short discussion on future works.

Keywords: color filter array, deep learning, demosaicing, hard thresholding, straight through estimator, gradient estimation

1. INTRODUCTION

In order to capture specific color information from visible light, digital color cameras require Color Filter Arrays (CFA).[1] CFAs are physical filters that facilitate the acquisition of a specific color channel per pixel. The CFAs currently found in digital cameras today are generally hand-designed patterns with different considerations depending on the camera's characteristics, perception characteristics of the human eye, and the environment and objective in which the camera is used.[1–4]

After receiving the reflected light from the environment through a CFA, the raw input corresponds to an image in which each pixel contains the intensity information of only one color and lacks the rest. The following process in image capture is called demosaicing, which estimates the unknown channel values in raw camera returns for each pixel.[5] There are various demosaicing algorithms, from simpler ones such as k-nearest-neighbor, bilinear, and bicubic interpolations[6,7] to more sophisticated demosaicing approaches for specific applications and different CFA patterns.[8–10] The papers[11,12] include detailed reviews of the classical demosaicing algorithms for various CFAs.

These hand-crafted CFAs might not exploit the actual features of natural images for a high-quality full-color image. Moreover, demosaicing may cause serious artifacts in the final image due to the lack of correlation information between the estimated channels and the collected ones. Each unique CFA design requires its own demosaicing algorithm depending on the filter pattern and available information from the captured scene, thus requiring prior knowledge of the captured information.

A few studies propose learning a CFA directly from natural images using deep learning (DL)-based methods.[13,14] These models learn CFAs that employ the full digital color spectrum; the learned filters acquire a linear combination of all color channels at each pixel as a single measurement. That is impractical in commercial cameras, where each pixel in a physical CFA observes a single color from a few available channels in the digital image. Some other studies assume working in the multispectral domain, and they learn multispectral filter arrays (MSFA).[15,16] Although the work on building commercial cameras with MSFAs in ongoing, these cameras are still in the research phase due to their high production cost, the available interpolation techniques for MSFA

Further author information: (Send correspondence to A.C.G.)
C.O.A.: E-mail: ca1389@msstate.edu,
A.C.G.: E-mail: gurbuz@msstate.edu

Real-Time Image Processing and Deep Learning 2024, edited by Nasser Kehtarnavaz, Mukul V. Shirvaikar,
Proc. of SPIE Vol. 13034, 130340M · © 2024 SPIE · 0277-786X · doi: 10.1117/12.3013594

filtered raw images are more prone to quality drops, and their raw image formats cost a lot more computation to get the usual RGB images as the final product.

This situation calls for a new, physically realizable CFA learning method that performs better than the hand-crafted filters. To the best of our knowledge, there is one method that learns a binary CFA using images in the RGBW configuration. This study[17] uses a set of weights for each pixel with a weight coefficient fed into a SoftMax function. The weight coefficient controls the output of SoftMax; larger coefficients give weights like a pseudo-binary mask.

In this study, we propose an alternative differentiable binary selection layer as a binary CFA module in a joint CFA learning-demosaicing framework. This joint framework is an end-to-end structure with two modules; the initial module learns a constrained binary CFA, and the second module reconstructs a color image from the CFA output. The learned CFAs can be directly implemented in physical camera array systems since the learned filters are physically applicable. The CFA learning module implements a hard thresholding operation compatible with backpropagation via gradient estimation.

The paper is structured in the following way. Section 2 describes the proposed CFA filter learning method and the whole training and testing procedure. Section 3 explains the hardware and software setup used in training and evaluation. Section 4 presents learned filters, obtained performance metrics and comparisons with the existing methods. Section 5 draws the conclusion and provides future work.

2. PROPOSED METHOD

We propose an end-to-end joint DNN architecture for simultaneous constrained filter learning and demosaicing. The proposed architecture has an initial binary channel selection layer and a demosaicing model based on available image reconstruction networks. This section explains each part of the proposed joint learning framework in detail.

2.1 The Joint Framework

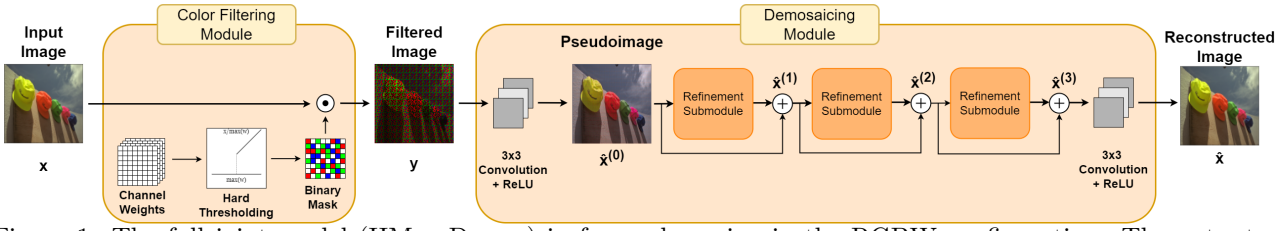


Figure 1: The full joint model (HMax-Demos) in forward passing in the RGBW configuration. The output of the HardMax represents the raw sensor input and is passed to the Demosaicing Module as input.

The joint framework is composed of two subnetworks; the binary CFA learning module and the demosaicing model. The training set consists of the $3P \times 3P \times C_{input}$ image blocks collected from digital images with a stride of P. The actual learnable CFA blocks have the $P \times P \times C_{input}$ number of learnable weights. After being initialized, the final $P \times P \times C_{input}$ binary filters are expanded by tiling operation to cover the $3P \times 3P \times C_{input}$ shape image, then $3P \times 3P \times 1$-size image measurement is acquired by masking operation and shrinking the channel dimension. This technique helps prevent reconstruction artifacts that appear at the edges of the reconstructed blocks. With a heuristic analysis, the P value is selected as 8, while the input channel number C_{input} depends on the test cases (3 for RGB, 4 for RGBW).

2.2 Binary CFA Learning Module

The proposed binary CFA learning module applies hard thresholding on a group of weights as a means of binary selection. In this module, the process begins with a set of weights of the same size as the expected full-color input image; for a full-color input image block with $P \times P$size and C_{input} number of channels, the binary CFA learning module has $P \times P \times C_{input}$ number of weights. For every pixel weight, the layer considers the index of the greatest weight value as the selected channel. This operation can also be achieved by weight normalization and hard thresholding operations, with the threshold value selected as 1. Thresholding the weights and replacing

the only non-zero weight with 1 (or dividing the thresholded value by its value) directly gives a digital mask that can be used as a CFA. The issue is that the thresholding operation is non-differentiable; therefore, it cannot be used in the learning scheme of neural networks.

Some methods approximate the hard thresholding operation with differentiable functions, but non-differentiable operations can be made directly applicable in gradient descent. A straight-through estimator (STE) is a gradient estimation method that involves applying the derivative of a differentiable function to a non-differentiable yet similarly behaving function.[18] Assume a function that is non-differentiable on some, or all, of its domain. We can observe the behavior of this function, find a differentiable function that behaves like the former function in the concerned interval, and assume that the gradient of this new function is the gradient of the former function during backpropagation. In the binary channel selection case, the hard thresholding operation is the non-differentiable part of the overall ML model. However, after adding normalization, the right side of the threshold is identical to the identity function. Therefore, the derivative of the hard thresholding is assumed as the derivative of the identity function during backpropagation. Figure 1 illustrates CFA learning with the proposed binary CFA learning approach.

2.3 Demosaicing Model

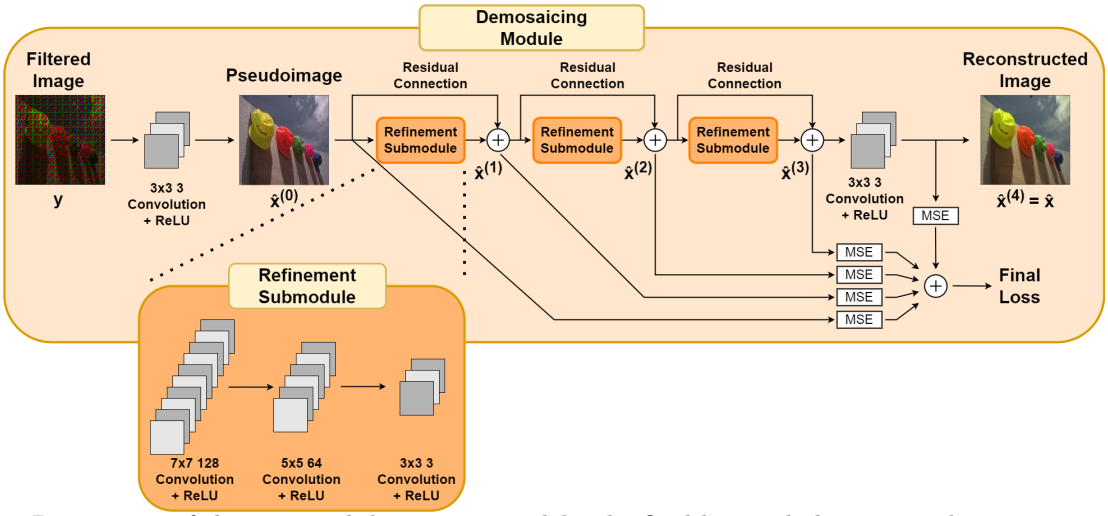


Figure 2: Description of the proposed demosaicer model. The final loss includes intermediate reconstructions.

The DL-based demosaicing module implemented in the joint CFA learning-demosaicing structure is designed based on the available works in demosaicing and image reconstruction models.[19,20] The input of the demosaicing model is the output of the CFA learning network; this represents the filtered raw camera measurements with the shape $3P \times 3P \times 1$. The output is a $3P \times 3P \times C_{image}$-sized reconstructed image block with values in intervals $[0, 255]$ corresponding to a full-color digital image. In this study, the reconstructed image is always in RGB configuration, therefore C_{image} is always 3. The reasoning behind using a nine times larger input patch area than the filter is to use the features around the patch intended for reconstruction to reinforce the demosaicing quality and prevent artifacts that occur around edges.

The pseudoimage is then sent into three consecutive refinement submodules, each of which is composed of three convolutional layers with ReLU activation functions following. The first layer has 128 7×7 kernels, the second layer has 64 5×5 kernels, and the third layer has 3 3×3 kernels. All layers apply padding to preserve the image size and include an L2 regularizer. There are skip connections between refinement submodules added to carry over the gradient. The model's final layer is another convolutional layer with 3 3×3 kernels and a ReLU activation function. The coarse reconstruction is guaranteed during training by including the pseudoimage and the consecutive refinement outputs in the loss function.

In the proposed architecture, the number of refinement submodules was selected as three to balance the model complexity and the image reconstruction quality.

2.4 Training Loss

The joint model's final loss function is defined as the sum of all mean squared error (MSE) loss computed over the total T training images for the final output, each refinement module outputs, and the pseudo image using the corresponding label image as given in (1).

$$\mathcal{L}_{total} = \sum_{i=1}^{T} \sum_{n=0}^{4} (x_i - \hat{x}_i^{(n)})^2, \tag{1}$$

where x_i represents the i-th training sample and $\hat{x}_i^{(n)}$ is the pseudoimage or the n-th refinement module output as shown in Fig.2. $\hat{x}_i^{(4)}$ denotes the final reconstructed image output of the model.

3. TRAINING SETUP

The training dataset was created from the 400 training and validation images from the BSDS500 dataset. BSDS500 is an image dataset very frequently found in demosaicing studies. The training image patches were created by fetching 24×24 patches with a stride value of 8 from the 400 training and validation images of the BSDS500 dataset. The grayscale values were added as the intensity channel for the RGBW case. In total, the training dataset consists of 881600 RGBW image patches.

Two different test datasets were used; 20 selected images from the test set of BSDS500, and the 24 images of the Kodak dataset were analyzed separately. The test images were divided into 24×24 patches with a stride value of 24. In the training scheme, the patches are filtered with a CFA, and then the filtered image patches are fed into the corresponding demosaicing model and finally merged the full-color reconstructions back into a full-scale image. The evaluation metrics were calculated with the full reconstructed image.

The training batch size for the models is 128, and the epoch number is 100. Jointly trained binary filtering and demosaicing models with a fixed learning rate stuck after a few epochs. To help the demosaicing model converge to a higher reconstruction performance, the learning step was exponentially decreased for each epoch, from 0.001 to 0.00001. The Adaptive Movement Estimation (ADAM) optimizer was used during backpropagation with decay rates (β_1 and β_2) and division constant ($\hat{\epsilon}$) values of ADAM optimizer being selected as 0.9, 0.999 and 10^{-7} respectively.

The code for the training and testing models is in Python 3. The TensorFlow and NumPy libraries were used for implementing the model. The tests were conducted in Python3 with Scikit-Image and Matplotlib libraries. All the models were trained in a dedicated machine with an Nvidia A6000 graphic card.

4. RESULTS

4.1 Evaluation

The reconstructed images are evaluated using the peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) metrics.

PSNR is used to observe the distortion of the reconstruction as additional noise compared to the original signal in terms of decibel units, and a higher PSNR value indicates a better reconstruction performance. PSNR can be computed as in (2) where the mean squared error (MSE) is computed over the whole reconstructed image. 255 is the highest value that a digital image channel can take.

$$PSNR = 20 \log_{10}(\frac{255}{MSE}) \tag{2}$$

SSIM is used to compare a reconstruction with a ground truth signal in terms of localized similarity, and it takes a value between 0 and 1, where a higher score indicates that the reconstructed image resembles the original image more. Calculation of SSIM involves the means and variances of the original (μ_x, σ_x^2) and the reconstructed ($\mu_{\hat{x}}$, $\sigma_{\hat{x}}^2$) images, along with the covariance of both ($\sigma_{x\hat{x}}$). The division stabilizer constants c_1 and c_2 in the SSIM definition in (3) are chosen as $(255 * 0.01)^2$ and $(255 * 0.03)^2$ respectively.

$$SSIM = \frac{(2\mu_x\mu_{\hat{x}} + c_1)(2\sigma_{x\hat{x}} + c_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + c_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + c_2)} \tag{3}$$

In order to evaluate the proposed CFA learning method, we devised four tests. The first test involves comparing the proposed demosaicing architecture with the alternative demosaicing architectures found in the literature. The second test looks at the effect of the CFA size on the image reconstruction performance of the joint network. The third test uses four common hand-crafted binary CFAs to compare their performance with the same demosaicing model as used in the proposed joint architecture. The fourth test compares the proposed joint CFA learning and demosaicing model with the available alternative.

4.2 Comparison of Demosaicing Models the Binary learned CFA

The first test looks into the performance of the proposed demosaicing model by comparing its performance with different demosaicing models presented in the studies that are found to be used in similar joint architectures. In this test, we selected three studies that have applied their own demosaicing models in joint CFA learning and demosaicing frameworks, implemented their demosaicing architectures with our proposed CFA learning module, and compared the images with respect to their PSNR and SSIM metrics.

Demosaicing Model		in[14]	in[17]	in[21]	Ours
Kodak	PSNR	38.321	37.738	32.252	**40.451**
	SSIM	97.671	97.142	93.663	**98.439**
BSDS500	PSNR	38.247	36.721	30.591	**40.054**
	SSIM	98.533	97.742	93.531	**98.967**

(a) RGB

Demosaicing Model		in[14]	in[17]	in[21]	Ours
Kodak	PSNR	40.632	37.959	33.359	**41.881**
	SSIM	98.629	97.385	94.578	**98.707**
BSDS500	PSNR	40.561	36.919	31.168	**41.181**
	SSIM	**99.188**	97.896	93.92	99.183

(b) RGBW

Table 1: Comparison of different demosaicing models used alongside the proposed CFA learning module for (a) RGB, and (b) RGBW configurations. Best performances for each metric and test dataset are shown with bold.

Table 1 shows the PSNR and SSIM performance of the compared demosaicers. The results show that our proposed demosaicer shows the highest score for both Kodak and BSDS500 datasets. In RGB case, our model's reconstructed images have 2.13 dB higher score for Kodak and 1.807 dB higher score for BSDS500 compared to the second highest model.

This test also shows that including the extra white channel improves the final image quality. The reconstruction quality of our model jumped by 1.43 dB for Kodak dataset and by 1.127 dB for BSDS500 dataset.

The two demosaicers, in[14] and our demosaicer, follows a simple feed-forward deep network structure compared to the models in[17] and,[21] which involve parallel architectures for processing different image information and merging them later in the model. On top of simple feedforward design, our model implements ideas from the image reconstruction studies,[22] which reinforces the demosaicing process as two objectives are similar in nature.

As we proved that our demosaicer performs higher compared to the other available alternatives, in the following sections, we will opt out for the proposed demosaicer model.

4.3 Analysis of the CFA Size

The second test involves training and testing joint models with different CFA sizes. For this test, four different CFA sizes were tested; 4×4, 8×8, 12×12, and 16×16. For each tested size, one RGB and one RGBW model were trained and evaluated.

CFA Size		4x4	8x8	12x12	16x16
Kodak	PSNR	38.561	**40.451**	39.535	38.865
	SSIM	97.77	**98.439**	98.236	98.125
BSD500	PSNR	37.57	**40.054**	38.676	37.880
	SSIM	98.26	**98.967**	98.726	98.558

(a) RGB

CFA Size		4x4	8x8	12x12	16x16
Kodak	PSNR	40.189	**41.881**	39.89	38.988
	SSIM	98.421	**98.707**	98.247	98.143
BSD500	PSNR	39.700	**41.181**	38.59	37.878
	SSIM	98.914	**99.183**	98.699	98.481

(b) RGBW

Table 2: Comparison of different CFA sizes for (a) RGB and (b) RGBW configurations.

Table 2 shows the results for the second test. The joint model with 8×8 size filters provide the highest image reconstruction performance. Any different size for CFA filters affect the final reconstruction performance negatively. The results also reinforce the idea that the extra white channel reinforces the reconstruction performance. Figure 3 shows the learned filters for this test.

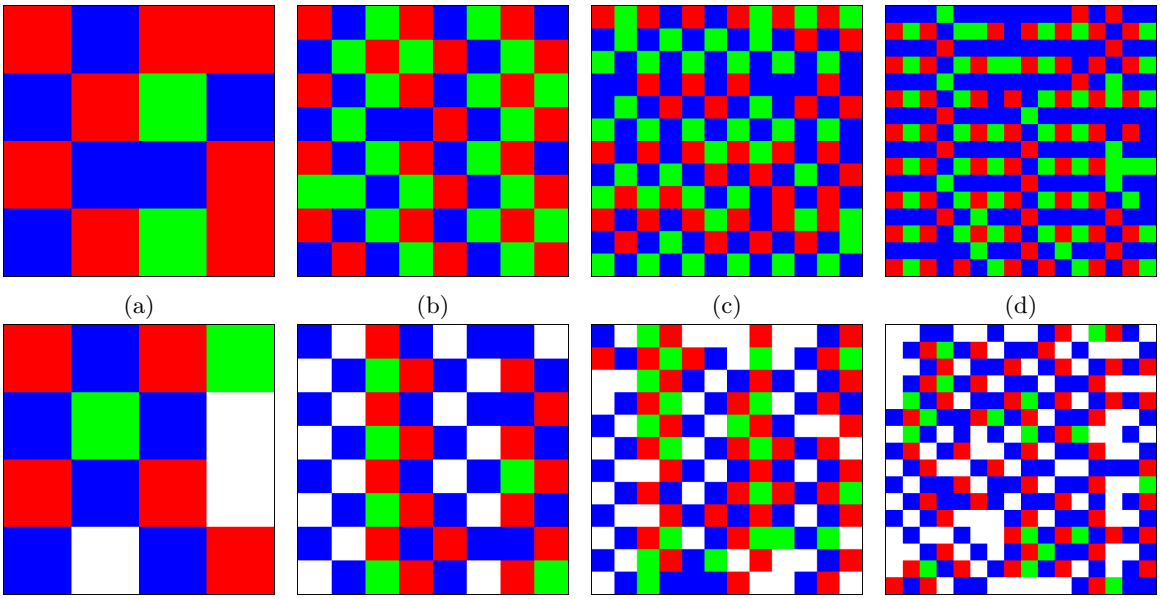
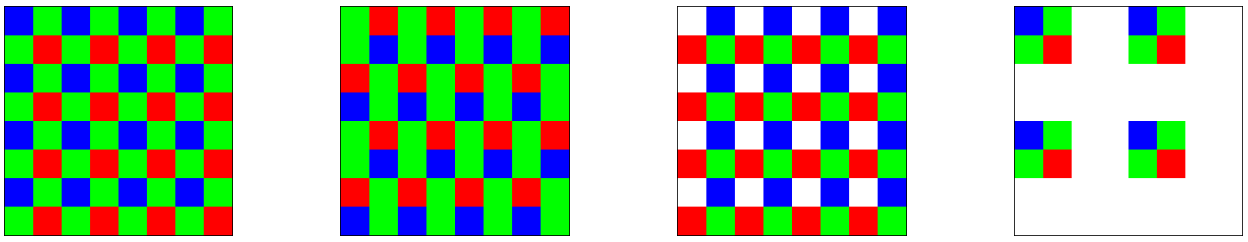


Figure 3: The learned CFAs with different sizes. (a) 4×4, (b) 8×8, (c) 12×12, (d) 16×16 filters for RGB configuration; (e) 4×4, (f) 8×8, (g) 12×12, (h) 16×16 filters for RGBW configuration.

4.4 Comparison of the Binary learned CFAs with Fixed CFAs

As we established the power of the proposed demosaicer and the optimal CFA size in the previous tests, in this test we aim to observe the performance of the proposed CFA learning module with respect to the traditional hand-crafted CFAs. For this test, we selected two RGB (Bayer[2] and Lukac[1]) and two RGBW (RGBW and CFZ[23]) CFAs, shown in Figure 4. All these hand-crafted CFAs were trained and evaluated separately with our proposed demosaicing architecture with exactly same parameters and random initialization.

Table 3 shows the quality of the reconstructed images. The learned binary CFA gives the highest PSNR



(a) (b) (c) (d)

Figure 4: Hand-crafted CFAs used in the evaluations: (a) Bayer, (b) Lukac, (c) RGBW, and (d) CFZ filters.

and SSIM scores in all comparisons. The average PSNR scores are 1.415 dB higher for Kodak and 0.47 dB higher for BSDS500 test images compared to the second highest filter (Bayer) in RGB case. In RGBW case, the difference is 0.83 dB for Kodak and 0.361 dB for BSDS500 test images compared to the second highest scoring filter (RGBW).

CFA		Bayer	Lukac	Proposed
Kodak	PSNR	39.036	38.683	**40.451**
	SSIM	97.857	98.083	**98.439**
BSD500	PSNR	39.584	38.897	**40.054**
	SSIM	98.769	98.732	**98.967**

(a) RGB

CFA		RGBW	CFZ	Proposed
Kodak	PSNR	41.051	38.643	**41.881**
	SSIM	98.657	98.093	**98.707**
BSD500	PSNR	40.820	37.981	**41.181**
	SSIM	99.131	98.592	**99.183**

(b) RGBW

Table 3: Comparison between the performance of the learned CFAs with the proposed module and hand-crafted CFAs with the same proposed demosaicing model. For (a) RGB and (b) RGBW configurations.

4.5 Comparison of the Binary learned CFAs with Alternative Learned CFAs

The final test compares the proposed binary CFA learning module with the only alternative model that was previously presented in.[17] The method in[17] implements a soft-thresholding algorithm based on SoftMax function. A set of randomly initialized weights for color channels for each pixel go through SoftMax operation controlled with a weight value. As the training progresses, this weight is increased to approximate the SoftMax output to binary mask.

CFA Learning Module		In[17]	Ours
Kodak	PSNR	39.034	40.451
	SSIM	97.816	98.439
BSD500	PSNR	37.819	40.054
	SSIM	96.373	98.967

(a)

CFA Learning Module		In[17]	Ours
Kodak	PSNR	39.607	41.881
	SSIM	97.884	98.707
BSD500	PSNR	38.474	41.181
	SSIM	98.387	99.183

(b)

Table 4: The comparison of performance between alternative learned CFAs. (a) for RGB configuration and (b) for RGBW configuration.

Table 4 shows the results. Our proposed binary CFA learning method bests the model presented in[17] by 1.417 dB in Kodak and by 2.235 dB in BSDS500 test images in the RGB case. In RGBW case, our model gives

2.274 dB higher score in Kodak and 2.707 dB higher in BSDS500 test images. We would like to note that the original study of[17] did not include evaluations for RGB case and compared their model only with their proposed hand-crafted CFA in.[23]

5. CONCLUSION AND FUTURE WORK

This study presents an introductory work on developing an efficient and physically applicable binary CFA learning method. The two primary goals of this work were to develop a method for learning CFAs that can be directly implemented in commercial digital cameras and to provide a better performance compared to the available hand-crafted filters and alternative methods. In this paper, we explained the background of the problem, introduced our method, and included the analyses to show the effectiveness of the proposed method.

The proposed architecture is designed to implement binary CFA learning for multiple problems. A further study may aim at learning binary CFAs for tasks different than reconstruction. One important example includes implementing optimal cameras for object detection to be used in autonomous vehicles. The flexibility of the overall proposed framework allows such modularity.

Future studies will include extensive analysis of the effects of different variables on the learned CFA and the performance, such as the training dataset size, variance in the training dataset, the number of epochs, the joint task, etc. The quality of the learned binary CFAs with the proposed method is an open question and further analysis of the learned patterns might help to develop better-informed CFAs.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant No. 2047771.

REFERENCES

[1] Lukac, R. and Plataniotis, K. N., "Color filter arrays: Design and performance analysis," *IEEE Transactions on Consumer electronics* **51**(4), 1260–1267 (2005).

[2] Bayer, B., "Color imaging array," *United States Patent, no. 3971065* (1976).

[3] Seiji, T., "Color imaging apparatus," *United States Patent, no. 8531563* (2013).

[4] Chung, K.-L., Chan, T.-H., and Chen, S.-N., "Effective three-stage demosaicking method for rgbw cfa images using the iterative error-compensation based approach," *Sensors* **20**(14), 3908 (2020).

[5] Gunturk, B. K., Glotzbach, J., Altunbasak, Y., Schafer, R. W., and Mersereau, R. M., "Demosaicking: color filter array interpolation," *IEEE Signal processing magazine* **22**(1), 44–54 (2005).

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

[7] Lukac, R., Plataniotis, K. N., Hatzinakos, D., and Aleksic, M., "A novel cost effective demosaicing approach," *IEEE Transactions on Consumer Electronics* **50**(1), 256–261 (2004).

[8] Kimmel, R., "Demosaicing: Image reconstruction from color ccd samples," *IEEE Transactions on image processing* **8**(9), 1221–1228 (1999).

[9] Li, X., "Demosaicing by successive approximation," *IEEE Transactions on Image Processing* **14**(3), 370–379 (2005).

[10] Lukac, R., Plataniotis, K. N., Hatzinakos, D., and Aleksic, M., "A new cfa interpolation framework," *Signal processing* **86**(7), 1559–1579 (2006).

[11] Li, X., Gunturk, B., and Zhang, L., "Image demosaicing: A systematic survey," in [*Visual Communications and Image Processing 2008*], **6822**, 489–503, SPIE (2008).

[12] Safna Asiq, M. and Sam Emmanuel, W., "Colour filter array demosaicking: a brief survey," *The Imaging Science Journal* **66**(8), 502–512 (2018).

[13] Tang, J., Li, J., and Tan, P., "Demosaicing by differentiable deep restoration," *Applied Sciences* **11**(4), 1649 (2021).

[14] Henz, B., Gastal, E. S., and Oliveira, M. M., "Deep joint design of color filter arrays and demosaicing," in [*Computer Graphics Forum*], **37**(2), 389–399, Wiley Online Library (2018).

[15] Bian, L., Wang, Y., and Zhang, J., "Generalized msfa engineering with structural and adaptive nonlocal demosaicing," *IEEE Transactions on Image Processing* **30**, 7867–7877 (2021).

[16] Zhang, F. and Bai, C., "Jointly learning spectral sensitivity functions and demosaicking via deep networks," in [*2021 3rd International Conference on Advances in Computer Technology, Information Science and Communication (CTISC)*], 404–411, IEEE (2021).

[17] Chakrabarti, A., "Learning sensor multiplexing design through back-propagation," *Advances in Neural Information Processing Systems* **29** (2016).

[18] Bengio, Y., Léonard, N., and Courville, A., "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432* (2013).

[19] Shi, W., Jiang, F., Liu, S., and Zhao, D., "Image compressed sensing using convolutional neural network," *IEEE Transactions on Image Processing* **29**, 375–388 (2019).

[20] Mdrafi, R. and Gurbuz, A. C., "Joint learning of measurement matrix and signal reconstruction via deep learning," *IEEE Transactions on Computational Imaging* **6**, 818–829 (2020).

[21] de Gioia, F. and Fanucci, L., "Data-driven convolutional model for digital color image demosaicing," *Applied Sciences* **11**(21), 9975 (2021).

[22] Ayna, C. Ö. and Gürbüz, A. C., "Robustness analysis for deep learning-based image reconstruction models," in [*2022 56th Asilomar Conference on Signals, Systems, and Computers*], 1428–1432, IEEE (2022).

[23] Chakrabarti, A., Freeman, W. T., and Zickler, T., "Rethinking color cameras," in [*2014 IEEE International Conference on Computational Photography (ICCP)*], 1–8, IEEE (2014).