A Deep Learning Approach to Mobile Camera Image Signal Processing

Jose Ivson S. Silva, Gabriel G. Carvalho, Marcel Santana Santos, Diego J. C. Santiago,
Lucas Pontes de Albuquerque, Jorge F. Puig Battle, Gabriel M. da Costa, Tsang Ing Ren

Centro de Informática (CIn), Universidade Federal de Pernambuco (UFPE), Brazil
Email: {jiss,ggc5,mss8,djcs,lpa,jfpb,gmc2,tir}@cin.ufpe.br

***Abstract*—The quality of the images obtained from mobile cameras has been an important feature for modern smartphones. The camera Image Signal Processing (ISP) is a significant procedure when generating high-quality images. However, the existing algorithms in the ISP pipeline need to be tuned according to the physical resources of the image capture, limiting the final image quality. This work aims at replacing the camera ISP pipeline with a deep learning model that can better generalize the procedure. A Deep Neural Network based on the UNet architecture was employed to process RAW images into RGB. Pre-processing stages were applied, and some resources for training were added incrementally. The results demonstrated that the test images were obtained efficiently, indicating that the replacement of traditional algorithms by deep models is indeed a promising path.**

I. Introduction

Nowadays, digital image processing and computer vision are employed in different areas ranging from entertainment, medical to industrial applications. Additionally, powerful mobile phones endowed with state-of-the-art cameras are released every year. They apply software-based solutions as an alternative for delivering a high-quality image since the hardware resources are restricted compared to high-end professional cameras. For non-professional purposes, the images captured by smartphones have quality comparable with standard cameras thanks to powerful image and signal processing procedures. As a consequence, smartphones have been taking the place of the traditional camera.

One key component in the camera image formation is the Image Signal Processing (ISP) pipeline, which transforms the raw data from the sensor to produce an image applying procedure to reduce noise and other artifacts, following by a color image processing pipeline [1]. These steps, which include *Black-level subtraction*, *Lens shading correction*, *White balance*, *Demosaic*, *Chroma denoise*, and *Color correction*, can be organized as the pipeline shown in Fig. 1. These algorithms and parameters, which depend on the sensor's characteristics, influence the final image's quality. Despite the advances, the ISP is still limited in mobile cameras due to the physical characteristics, such as small sensors and compact lenses, that restrict the focal distance. Further, to achieve a satisfactory quality, a tuning procedure is still necessary for a new device production.

In this work, we propose a Deep Neural networks (DNNs) architecture based on UNet [2] to process the raw data and produce a color processed image. Through this approach, the tuning can be generalized for new devices saving manual efforts from image quality experts, besides being more efficient than methods based on automatic ISP algorithms [3]. During training, we employ the loss functions used in style transfer applications [4] to capture global information like brightness and contrast.

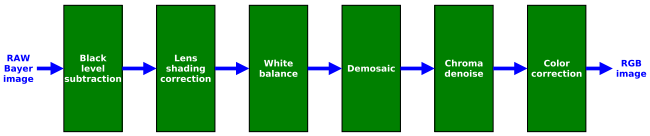


Fig. 1. Traditional camera ISP pipeline

This paper is organized as follows: In Sec. II we point recent and relevant works concerning the processing of RAW images into RGB images. In Sec. III we shed some details of the pre-processing stage. Then, in Sec. IV, some details concerning the architecture of the neural network are presented. In Sec. V we show the results of our experiment. Finally, in Sec. VI, we summarize the work and point some of the possible extensions.

II. Related Works

HDR+ [5] captures a burst of images in different exposures to be merged before running ISP and tone mapping steps. The project provides a dataset containing the burst, and the final result that is used to train our network. The idea of High Dynamic Range (HDR) images [6] is that it increases the bit depth of the result and applies a tone mapping, converting the result to a range allowed by the display in the device, and it is applied to improve the visualization when the image in processing has light and dark areas causing results with overexposed and dark regions. Both [5] and [6] deal with image processing using classical algorithms.

Conversely, Deep Neural networks (DNNs) have been used in image-to-image tasks [7], [8] and it is expected for a reasonable good network to learn this processing, assuming a decent amount of data, without any concern to find the best algorithm and parameters for each step. HDRNet [9], for instance, performs the enhancement in real-time with a convolutional architecture. Part of this architecture uses a low-resolution image copy to extract local and global features. DeepISP [10]

was proposed with a fully convolutional network with a stage to extract low-level features for local modifications and another stage to high-level features for global correction. The training and evaluation are performed in image pairs from the same smartphone model. More recently, PyNET [11] was proposed with a pyramidal CNN architecture that learns the ISP steps. It processes the image in different scales combining global and local features. Further, the work provided a dataset containing 10 thousand full-resolution RAW-RGB image pairs.

III. Pre-processing Stages

In this section, we describe the relevant steps in our pre-processing stage. All of these procedures are applied to the RAW images before feeding them to the network. In our approach we use the images from the Google HDR+ dataset [5].

A. Fixing and Converting the Dataset

The initial treatment consists of rotating and cropping to ensure that both the input RAW image (merged.dng) and the ground truth image have consistent shapes and are aligned (an issue found in Google's HDR+ dataset). Subsequently, we convert the RAW merged file from DNG to a $16-$bit PNG file, in which we include the following procedures.

B. Black-Level Subtraction

Black-Level subtraction is a procedure to deduct an offset from all pixels so that pixels receiving no light become zero. The sensor-dependent values used for the black-level and white-level are read from the RAW files, and the channel-wise transformation is summarized by the equation

$$Im_{\text{bls}}(x, y, c) = \frac{Im(x, y, c) - bl(c)}{wl}, \tag{1}$$

where bl and wl are the black and white levels, respectively, c represents the channel in which the operation is performed, Im is the input image, and Im_{bls} is the resulting image after black-level subtraction.

C. Lens Shading Correction

Lens shading correction is a procedure to brighten the corners of the image, compensating for lens vignetting [12] and correcting spatially varying color due to light hitting the sensor at an oblique angle. The Google HDR+ dataset contains lens shading masks for each capture in the burst. We select the mask corresponding to the reference frame and resize it. The lens shading correction is obtained by the elementwise product (array product)

$$Im_{\text{lsc}}(x, y, c) = Im(x, y, c) \cdot LSM(x, y, c), \tag{2}$$

where c represents the channel, $LSM(x, y, c)$ is the resized lens shading mask, Im is the input image, and Im_{lsc} is the resulting image after lens shading correction.

D. White Balance

White balancing [13] linearly scales the four RGGB channels so that grays in the scene map to grays in the image. Likewise, we use values recorded in the metadata contained in the RAW files.

$$Im_{\text{wb}}(x, y, c) = Im(x, y, c) \cdot wb(c), \tag{3}$$

where $wb(c)$ is the white-balance value for channel c, Im is the input image, and Im_{wb} is the resulting image after white balance.

E. Cropping the dataset

Having the pre-processed files, and the ground truth images provided by the Google HDR+ project, we crop the whole dataset in small patches of size 448×448 that are used to train/test. The dataset, after cropping, consists of approximately 2000 patches for validation, 2000 patches for testing, and 82000 patches for training.

IV. Approach

Our goal is to synthesize a high-quality RGB image from the RAW measurements of the camera sensor by replacing the traditional ISP pipeline by a single convolutional neural network (CNN). This network takes the RAW Bayer data and produces the final RGB image. In this section, we discuss our proposed network architecture and loss function for training our model.

A. Modified UNet Architecture

There are many possible choices for architectures when building convolutional networks. In [11], the authors proposes a network to process RAW images and output RGB images. However, running that network is very costly and slow. Due to requirement of deploying our model on smartphones devices and the image-to-image nature of our problem, we decided to use the UNet [2] architecture. This architecture as lighter when compared to the PyNet [11] and still is able to produce high-quality results. The UNet has a sequence of double convolutional layers in two stages. In the downstage, after each double convolutional layer, it is applied a max pooling. In the upstage, after each double convolutional layer, it is applied an upsampling. The layers of downstage are skip connected with the upstage. This allow the information to go forward before the downscale made in the max-pooling layer. In the end, the pixel convolutional layer with kernel size 1×1 reduces the dimensionality for 3 output channels corresponding to RGB representation. All of the convolutional layers are followed by a *ReLU* function.

The UNet originally receives an input with one channel that represents medical data and outputs a mask for the segmentation. Since, in our case, the input is a RAW Bayer image in CFA pattern [14] and the output is an RGB image, the input is deinterleaved in four channels (RGGB) before the first convolutional layer. That way, the role of the first UNet layer, receiving just one channel, is not drastically altered (See Fig. 2). This reduces the height and width by a factor of 2,

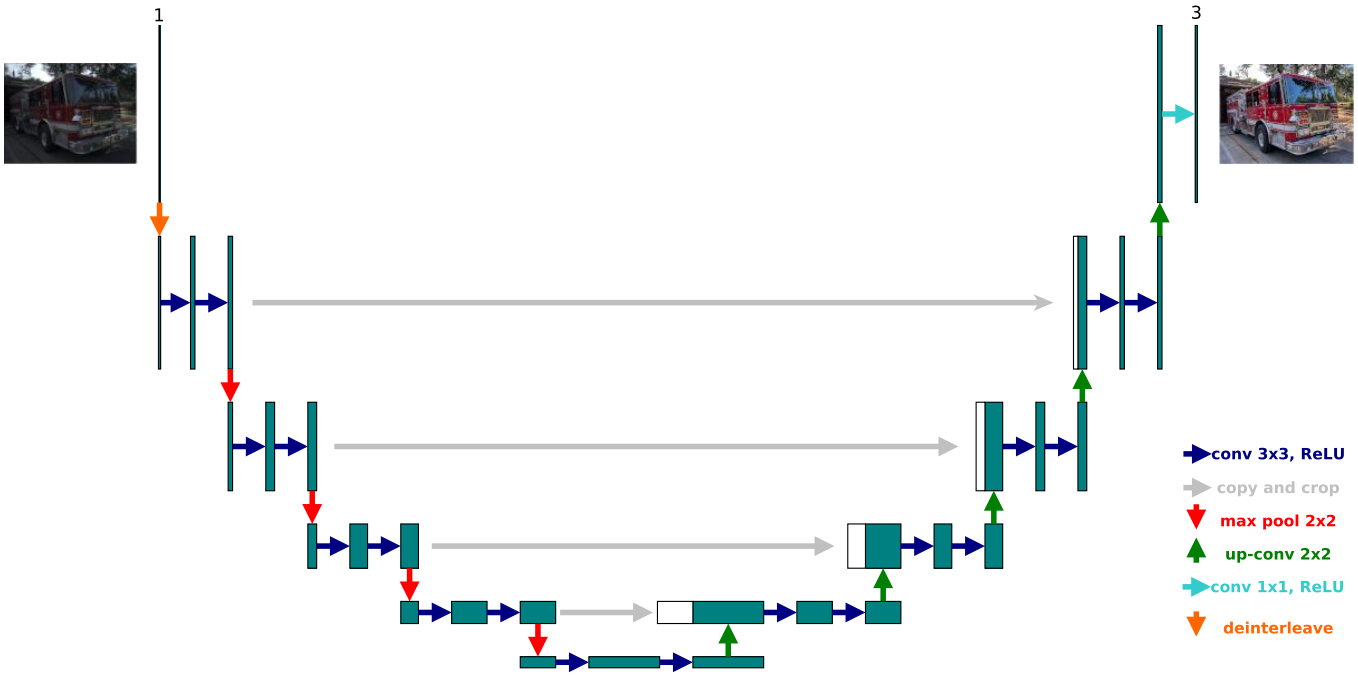


Fig. 2. Architecture based on UNet. The input is a RAW Bayer image and the output is RGB.

thus needing an additional upsample to restore the original dimensions.

B. Loss Function

The loss function gives a measure of how far is the network output $\hat{I}$ to the ground truth I, steering the manner the optimizer updates the network's weights. The choice of the loss function is critical in each learning system. Our goal is to synthesize images high visual quality, compared or superior to the conventional ISP pipelines. Ideally, the loss function must capture perceptually important differences between the estimated and reference images. Inspired by recent image-to-image approaches [4], [15], [16], we train our network using a perceptual loss function. Specifically, our loss function is a combination of the MSE loss L_{MSE} and a perceptual measure which is obtained by the VGG loss L_{VGG}, style loss L_{Style}, and total variation L_{TV} regularization [4], as follows:

$$L = \lambda_1 L_{MSE} + \lambda_2 L_{VGG} + \lambda_3 L_{Style} + \lambda_4 L_{TV} \tag{4}$$

where $\lambda_1 = 1.0$, $\lambda_2 = 0.8$, $\lambda_3 = 120.0$ and $\lambda_4 = 0.1$ in our implementation.

a) MSE Loss: Encourages local similarity by computing the pixel-wise l_2 difference between the output and the ground-truth, as follows:

$$L_{MSE} = \|I - \hat{I}\|_2. \tag{5}$$

b) VGG Loss: To encourage perceptual similarity between the output image and the ground-truth, we leverage the VGG loss. To compute this term, we use the pre-trained VGG-19 [17] to obtain features from network output $\hat{I}$ and the ground truth I and then compute l_2 between them, as follows:

$$L_{VGG} = \left\| \hat{I}^{VGG} - I^{VGG} \right\|_2 \tag{6}$$

where $\hat{I}^{VGG}$ and I^{VGG} are the VGG-19 features of the network output and ground-truth respectively.

c) Style Loss: Measures the differences in style (*e.g.*, colors, textures, etc) by comparing global statistics with a Gram matrix. [18], [19] propose the style loss defined as:

$$L_{Style} = \left\| G\left(\hat{I}^{VGG}\right) - G\left(I^{VGG}\right) \right\|_1 \tag{7}$$

where $G(.)$ is the Gram matrix [20].

d) Total Variation: To induce spatial smoothness in the network output $\hat{I}$ we follow prior work [4], [16] and use the total variation regularizer [21] defined as follows:

$$L_{TV} = \frac{1}{N} \sum_{x,y,c} \left(\hat{I}(x, y+1, c) - \hat{I}(x, y, c) \right) + \left(\hat{I}(x+1, y, c) - \hat{I}(x, y, c) \right), \tag{8}$$

where N is the number of elements in $\hat{I}$.

C. Implementation

a) Group Normalization: Both the input image and the hidden features can be normalized. The Group Normalization [22] normalizes the distribution of a group of channels along

the height and width dimensions by the same mean and standard deviation. It learns the scale and shift of a linear projection of the normalized feature. As pointed out in [22], Group Normalization is stable under batch size variations, whereas Batch Normalization [23] yields large errors when the batch size is small. Therefore, for image-to-image applications, where memory plays a significant role and batch sizes should be small, Group Normalization is recommended. Therefore, unless stated the contrary, we use Group Normalization within each layer of our model.

b) Optimization: We initialize our network using the Xavier approach [24]. We train the network during 50 epochs with a learning rate of 0.001. We use the Adam optimizer with AMSGrad [25] enabled and its default parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and mini-batch size of 8.

V. Experiments

A. Training Data

We use the Google HDR+ dataset [5] for training with the split described in Sec. III-E. This dataset contains images taken using several devices in a myriad of scenarios (Fig. 3), which is an important factor to avoid overfitting and allow generalization. The validation set is used to get the weights from the epoch of minimum loss, which we use to the test images.

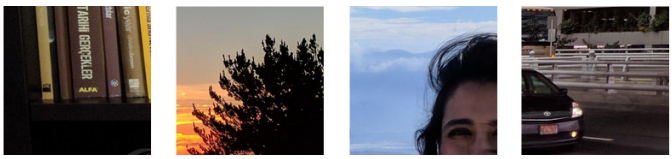


Fig. 3. Samples patches from HDR+ dataset used to train the network

B. Test

For testing our method we captured several images from a mobile camera in different scenarios, such as indoors, outdoors, different illumination settings, etc. The device model used for capturing these test images is not included in the list of devices used in the training dataset. Finally, 50 images were selected to evaluate the results in terms of the SSIM and RMSE quality metrics [26], [27].

C. Results

For the best network configuration we also display the absolute difference between the ground truth and our output. Fig. 4 shows an example of a RAW/ground-truth pair of a picture taken against the Sunlight. Next, we show the incremental results of the network adding resources for training the proposed architecture.

The first training configuration of the network included an Adam optimizer, MSE and VGG losses. In this setting we did not use the pre-processing described in Sec. III. Instead, we divide the input RAW data by its bit depth value. As we can see in Fig. 5, with this setup the network produces results with color shift on the sky.

(a) RAW

(b) Ground truth

Fig. 4. Example of RAW and ground-truth images

Fig. 5. The church image result with Adam optimizer, MSE and VGG losses, and bit-depth normalization

The metadata read from the RAW files provide values for the black-level, white balance, and lens shading. This information is used to normalize the data as described in Sec. III (Fig. 6). We adopt the metadata normalization as default since it fixes an observed red casting caused by the bit-depth normalization.

Fig. 6. Metadata normalization of input raw data

The style loss and total variation regularization were incorporated into training (Fig. 7). These functions were introduced to bring global information, although in some cases it put blemishes in some regions of the processed image.

Fig. 7. Result of training with style loss and total variation regularization

The AMSGrad was enabled in the Adam optimizer reducing some artifacts introduced by the loss used before (Fig. 8).

Fig. 8. Result of training with AMSGrad optimizer

The group normalization is used with 16 channels per group, resulting in an image with enhanced contrast (Fig. 9). The SSIM and RMSE when comparing with the ground truth (Fig. 4) are 0.881 and 0.0497, respectively.

In day light conditions (Fig. 10 and Fig. 11), the network result was satisfactory and very similar to the reference with high SSIM and low RMSE values.

In low-light conditions (Fig. 12 and Fig. 13), the device processing brightens the image and as a drawback adds noise in the resulting image. The dissimilarity obtained by the metrics is justified by the device employing an algorithm to automatically enhance the illumination in dark scenes. Nonetheless, the net result has more fidelity to the real scene and no significant noise.

The distribution of SSIM (Fig. 14) is in high values and RMSE (Fig. 15) in low values indicating a strong correlation with the reference for most test images. The image with the highest SSIM in the test dataset is a landscape with the sunset (Fig. 16). And the lowest RMSE is an indoor image with good illumination (Fig. 17).

VI. Conclusion

This work proposed a deep neural network architecture to process RAW into RGB images, replacing the traditional

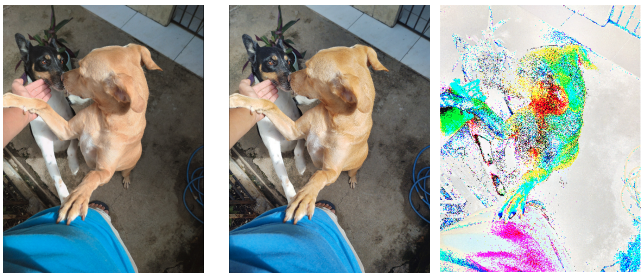
(a) Network output

(b) Difference from ground-truth

Fig. 9. Result using group normalization. SSIM=0.881 and RMSE=0.0497.

(a) Network output (b) Ground-truth (c) Difference

Fig. 10. Leaves in direct sunlight. SSIM=0.842 and RMSE=0.0485.



(a) Network output (b) Ground-truth (c) Difference

Fig. 11. Outdoor capture in good light conditions. SSIM=0.8375 and RMSE=0.0817.

ISP steps. The resulting images were successfully processed, exhibiting reasonable perceptual quality, SSIM, and RMSE when compared with their corresponding ground truth images. It was observed that the best results were obtained when processing scenes with good illumination. The results indicate

(a) Network output (b) Ground-truth (c) Difference

Fig. 12. Low light scenario: picture of the Moon. SSIM=0.3123 and RMSE=0.1892.

(a) Network output (b) Ground-truth (c) Difference

Fig. 13. Low light scenario: picture of a plant with landscape. SSIM=0.303 and RMSE=0.157.

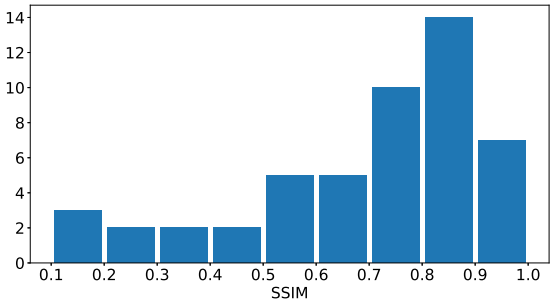


Fig. 14. SSIM distribution

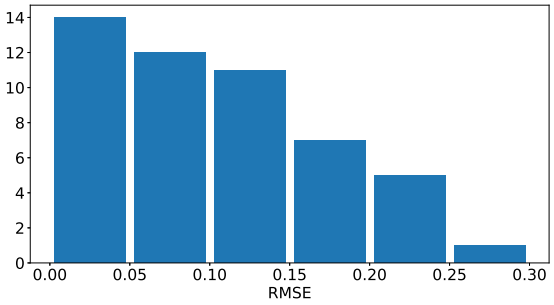


Fig. 15. RMSE distribution

that the deep learning model can be employed to replace a traditional ISP, with possible generalization for other devices without fine-tuning the parameters.

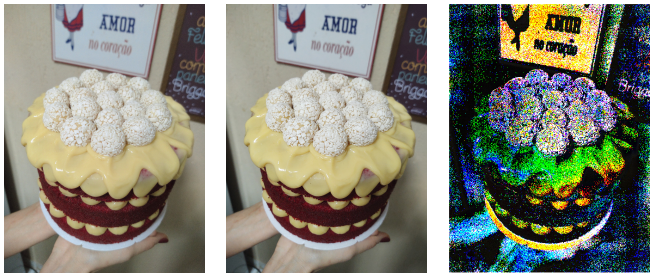
To yield better results, the network can be improved by including global components in its architecture, as well as the usage of data augmentation to include different types of camera sensors into the training data.

(a) Network output

(b) Ground-truth

(c) Difference

Fig. 16. Landscape scenario: picture of the sunset. SSIM=0.946 and RMSE=0.0303.



(a) Network output (b) Ground-truth (c) Difference

Fig. 17. Indoor capture with good light conditions. SSIM=0.9005 and RMSE=0.03.

ACKNOWLEDGMENT

This work was supported by the research cooperation project between Motorola Mobility (a Lenovo Company) and CIn-UFPE. We also acknowledge support from CNPQ and CAPES Brazilian governmental agencies.

REFERENCES

[1] R. Ramanath, W. E. Snyder, Y. Yoo, and M. S. Drew, "Color image processing pipeline," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 34–43, 2005.

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

[3] J. Nishimura, T. Gerasimow, R. Sushma, A. Sutic, C.-T. Wu, and G. Michael, "Automatic isp image quality tuning using nonlinear optimization," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 2471–2475.

[4] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*. Springer, 2016, pp. 694–711.

[5] S. W. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. T. Barron, F. Kainz, J. Chen, and M. Levoy, "Burst photography for high dynamic range and low-light imaging on mobile cameras," *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, vol. 35, no. 6, 2016.

[6] E. Reinhard, W. Heidrich, P. Debevec, S. Pattanaik, G. Ward, and K. Myszkowski, *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, 2010.

[7] S. Kaji and S. Kida, "Overview of image-to-image translation by use of deep neural networks: denoising, super-resolution, modality conversion, and reconstruction in medical imaging," *Radiological Physics and Technology*, vol. 12, no. 3, pp. 235–248, 2019.

[8] Z. Yan, H. Zhang, B. Wang, S. Paris, and Y. Yu, "Automatic photo adjustment using deep neural networks," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 2, pp. 1–15, 2016.

[9] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, and F. Durand, "Deep bilateral learning for real-time image enhancement," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–12, 2017.

[10] E. Schwartz, R. Giryes, and A. M. Bronstein, "Deepisp: Toward learning an end-to-end image processing pipeline," *IEEE Transactions on Image Processing*, vol. 28, no. 2, pp. 912–923, 2018.

[11] A. Ignatov, L. Van Gool, and R. Timofte, "Replacing mobile camera isp with a single deep learning model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 536–537.

[12] R. Lukac, *Computational Photography: Methods and Applications*, ser. Digital Imaging and Computer Vision. CRC Press, 2010.

[13] ——, *Single-sensor imaging: methods and applications for digital cameras*. CRC Press, 2018.

[14] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau, "Demosaicking: color filter array interpolation," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 44–54, 2005.

[15] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1511–1520.

[16] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 85–100.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[18] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 262–270.

[19] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.

[20] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge University Press, 2012.

[21] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5188–5196.

[22] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 448–456.

[24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010, pp. 249–256.

[25] S. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *International Conference on Learning Representations*, 2018.

[26] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *2010 20th International Conference on Pattern Recognition*. IEEE, 2010, pp. 2366–2369.

[27] T. Chai and R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)?–arguments against avoiding rmse in the literature," *Geoscientific Model Development*, vol. 7, no. 3, pp. 1247–1250, 2014.