Learning Raw Image Denoising with Bayer Pattern Unification and Bayer Preserving Augmentation

Jiaming Liu[1] Chi-Hao Wu[1] Yuzhi Wang[2] Qin Xu[1] Yuqian Zhou[1] Haibin Huang[1]
Chuan Wang[1] Shaofan Cai[1] Yifan Ding[3,1] Haoqiang Fan[1] Jue Wang[1]
[1]Megvii Technology [2]Tsinghua University [3]University of Central Florida
[1]{liujiaming,wuqihao,xuqin,zhouyuqian,huanghaibin,wangchuan,caishaofan,fhq,wangjue}@megvii.com
[2]yuzhiwang@tsinghua.edu.cn [3]yf.ding@knights.ucf.edu

Abstract

In this paper, we present new data pre-processing and augmentation techniques for DNN-based raw image denoising. Compared with traditional RGB image denoising, performing this task on direct camera sensor readings presents new challenges such as how to effectively handle various Bayer patterns from different data sources, and subsequently how to perform valid data augmentation with raw images. To address the first problem, we propose a Bayer pattern unification (BayerUnify) method to unify different Bayer patterns. This allows us to fully utilize a heterogeneous dataset to train a single denoising model instead of training one model for each pattern. Furthermore, while it is essential to augment the dataset to improve model generalization and performance, we discovered that it is error-prone to modify raw images by adapting augmentation methods designed for RGB images. Towards this end, we present a Bayer preserving augmentation (BayerAug) method as an effective approach for raw image augmentation. Combining these data processing technqiues with a modified U-Net, our method achieves a PSNR of 52.11 and a SSIM of 0.9969 in NTIRE 2019 Real Image Denoising Challenge, demonstrating the state-of-the-art performance.

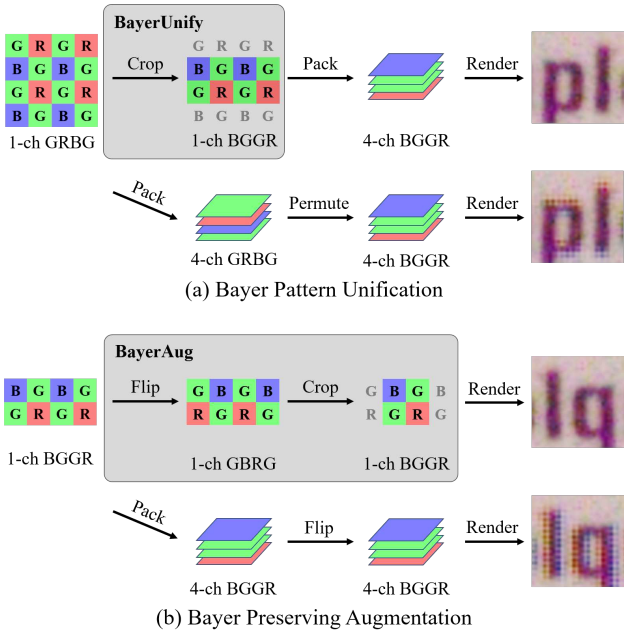


Figure 1. Demonstration of our proposed (a) Bayer pattern unification and (b) Bayer preserving augmentation. Our method unifies and augments the Bayer raw images without affecting the content, while improper pre-processing or augmentation would disturb the spatial relationship of the raw images and therefore result in artifacts.

1. Introduction

Image denoising is one of the fundamental problems in image processing and computer vision, and restoring high quality images from extremely noisy ones remains to be challenging. This can be even worse when it comes to images taken from mobile devices. Due to the use of relatively low-cost sensors and lenses, images captured by mobile cameras can be severely corrupted by high level noise, especially in low-light scenarios. Many denoising methods have been proposed to address this problem, including traditional methods such as NLM [6] and BM3D [10] as well as more recent deep neural network (DNN) based denoising models [26, 9, 31, 19, 28, 23, 27], but their performances are still far from satisfactory on mobile devices.

Recently, thanks to the public raw image denoising datasets [1, 8, 4], denoising raw image data has received more and more attentions and has shown promising results

This work is supported by The National Key Research and Development Program of China (2018YFC0831700).

[8, 17, 12]. Raw images are direct readings from images sensors, with camera filter arrays (CFAs) arranged in specific patterns such as the Bayer pattern [5]. These digital signals are further post-processed to obtain RGB images through a complex pipeline including lens shading correction, white balancing, demosaicking, gamma correction, *etc*. [14]. Therefore, original noise properties that exist in raw images are often distorted in RGB images, making the noise harder to remove afterwards. This means that there are potentially better denoising methods that can be developed on the raw image data [8], compared with many works done in RGB domain. In this work, we study the problem of raw image denoising using DNN, and our focus is on how to train an effective raw image denoising model by proper data pre-processing and data augmentation.

First of all, to perform raw image denoising with DNN models, it is a common practice to pack a Bayer raw image into a 4-channel RGGB image, and feed it into neural networks [8]. With data collected from cameras with different Bayer patterns, a simple solution is to train one model for each pattern. However, this decreases the size of the effective training set and thereby hurts the performance. To fully utilize all training data to achieve better performance, we propose a Bayer pattern unification (BayerUnify) technique to eliminate the differences among Bayer patterns. As illustrated in Fig. 1 (a), flipping and cropping operations are employed to turn a specific CFA pattern into another one, with which we can unify all training images into the same pattern. As a result, all the training data can be used together to optimize a single model to achieve the best possible result.

Data augmentation is a common approach in deep learning to improve model performance by increasing the diversity of a training dataset. However, data augmentation of raw images is not as straightforward as that of RGB images. An example in shown in Fig. 1 (b). Simply flipping the packed 4-channel raw images is erroneous because it results in an image that is impossible in real world. This phenomenon can also be found in other types of augmentation operations such as cropping, transposition, *etc*. To tackle this problem, we introduce a Bayer preserving augmentation (BayerAug) technique that allows proper augmentation for raw images. As shown in Fig. 1 (b), extra operations are required to correctly flip a raw image.

Both BayerUnify and BayerAug techniques are simple, yet effective ways for increasing the training data size and diversity for raw image denoising. We apply these techniques to train models based on our modified U-Net [25], and achieve state-of-the-art results in NTIRE 2019 Real Image Denoising Challenge (Track 1) [2]. We conduct detailed experiments and analysis in Sec. 4 to validate the proposed techniques.

To summarize, our contributions include:

- We propose a novel Bayer pattern unification technique (BayerUnify) to transform different Bayer patterns into an unified one, enabling training and inference of a single denoising model using data collected from different digital cameras.

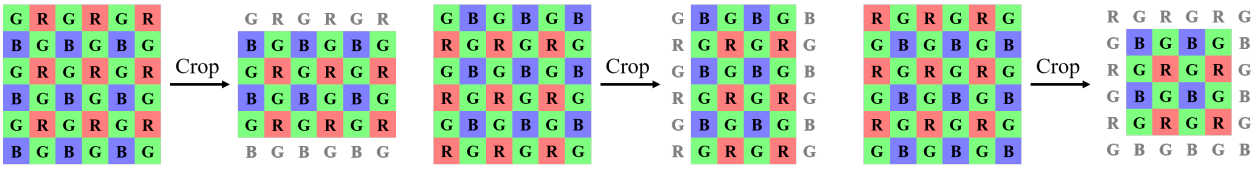
- We introduce a Bayer preserving augmentation technique (BayerAug) to allow effective augmentation for raw images for further performance improvement.

- We build DNN models based on U-Net for raw image denoising, and apply the above techniques to achieve the best results in NTIRE 2019 Real Image Denoising Challenge (Track 1). Extensive experiments are conducted to demonstrate the effectiveness of proposed method.

2. Related Works

Image desnoising is a fundamental low-level image processing problem that has been studied for decades. Classical approaches include non-local means (NLM) [6], sparse coding [11, 22, 3], 3D transform-domain filtering (BM3D) [10], and others [13, 24]. Image priors are often required to design these methods. In recent years, convolutional neural networks (CNN) allows end-to-end training of discriminative models that have achieved great success in this field. As an earlier work, [7] applied multi-layer perceptron (MLP) to achieve comparable results with BM3D. Later on, more advanced network architectures are proposed. For example, [29] uses residual learning and batch normalization to achieve great performance improvement. [26] applies memory blocks to tackle the long-term dependency problems in previous CNN architectures. A large number of CNN works are continuously proposed [9, 19, 28, 23, 27, 30, 20] to tackle this long-standing problem.

While there are many single image denoising approaches focusing on RGB images, raw image denoising has attracted much less attention due to the lack of training data. Some works use CNN to process the noisy raw images to produce clean RGB images [8, 17, 12] and achieve impressive visual results. In these works, the CNN models not only deal with the denoising problem, but concurrently handle other problems such as demosaicking and so on. For pure raw to raw image denoising, it is feasible to directly apply RGB denoising methods on raw images as demonstrated in [1], but the performance is limited. Recently, with more publicly available raw image datasets [1, 8, 4], it is expected to see more efforts put on solving the raw image denoising problem. In this paper, we mainly resolve the specific issues of data pre-processing and augmentation with Bayer raw images.



(a) GRBG to BGGR (b) GBRG to BGGR (c) RGGB to BGGR

Figure 2. Unify Bayer pattern via cropping in the training phase.

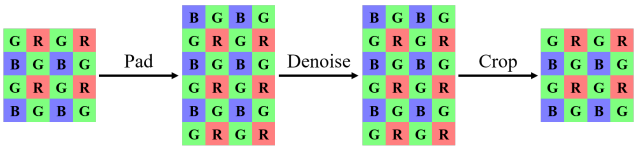


Figure 3. Unify Bayer pattern via padding and disunify via cropping in the testing phase.

3. Proposed Method

3.1. Bayer Pattern Unification (BayerUnify)

The Bayer patterns of raw images fall into different categories. To apply a single CNN to denoise raw images with different Bayer patterns, it is essential to align the order of the channels since different channels capture different regions of wavelength. In the meantime, the structural information laid in adjacent pixels from different channels has to be maintained. Based on these principles, we propose multiple ways to convert a raw image from one Bayer pattern to another, which are applicable to different scenarios.

3.1.1 Training: Unify via Cropping

In the training stage, we unify raw images with different Bayer patterns via cropping. By scarifying a minor number of pixels, it enables us to use raw images from different cameras to train a single denoising model, and thereby increases the number of available training samples.

We first introduce our notations of Bayer patterns. We represent each pattern by the sequence of its channels within each 2×2 block, in the order of top-left, top-right, bottom-left, and bottom-right. Typically, there are 4 possible formats, namely RGGB, BGGR, GRBG, and GBRG. For clarity, we use BGGR as the target format to illustrate our method.

Cropping odd number of rows or columns creates offsets which alter the Bayer pattern. As shown in Fig. 2, cropping the first row and the last row changes a $C_1C_2C_3C_4$ image into a $C_3C_4C_1C_2$ image (e.g. GRBG to BGGR). Likewise, cropping the first and the last column alters $C_1C_2C_3C_4$ into $C_2C_1C_4C_3$ (e.g. GBRG to BGGR). These two operations together convert $C_1C_2C_3C_4$ into $C_4C_3C_2C_1$ (e.g. RGGB to BGGR). Hence, one can normalize any Bayer pattern to an unified one by cropping.

3.1.2 Testing: Unify via Padding, Disunify via Cropping

We have shown that one can train a Bayer-pattern-specific network with raw images of different patterns. Moreover, it is possible to denoise images of different patterns with the trained network. Due to the fact that every pixel of the input images needs to be processed, instead of cropping some pixels from the input images, we unify their Bayer patterns via padding some pixels. After network denoising, we simply remove these extra pixels to disunify the output images. This process is illustrated in Fig. 3.

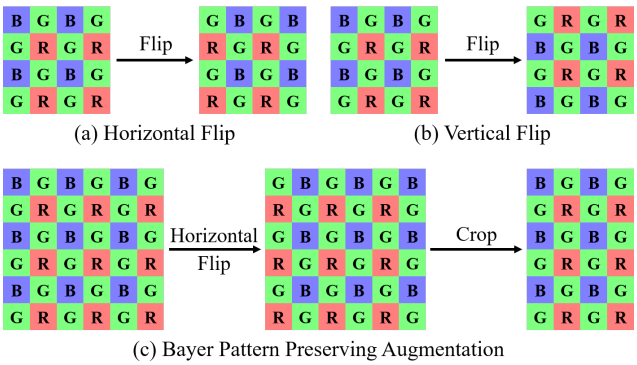
Padding alters the Bayer pattern in a similar way to cropping. Padding one row of pixels to the top and the bottom changes a $C_1C_2C_3C_4$ image into a $C_3C_4C_1C_2$ image (e.g. GRBG to BGGR); padding one column to the left and to the right turns $C_1C_2C_3C_4$ into $C_2C_1C_4C_3$ (e.g. GBRG to BGGR); padding to all four edges transforms $C_1C_2C_3C_4$ into $C_4C_3C_2C_1$ (e.g. RGGB to BGGR).

Hence, we can apply padding to convert any pattern to the desired one. As a straightforward disunification, removing the padded pixels reverses this conversion. Note that we apply reflection padding (aka "reflect" for numpy.pad or "BORDER_REFLECT_101" for OpenCV) to make sure the additional pixels come from the correct channel.

3.2. Bayer Preserving Augmentation (BayerAug)

When training a neural network for vision and graphic tasks on RGB images, it is common to apply flipping and cropping as data augmentation methods. They increases the effective number of samples dramatically while being very concise. However, for Bayer raw images, flipping operations may affect the Bayer pattern. As illustrated in Fig. 4 (a) and (b), a horizontal flip switches the Bayer pattern from $C_1C_2C_3C_4$ to $C_2C_1C_4C_3$, and a vertical one switches the pattern from $C_1C_2C_3C_4$ to $C_3C_4C_1C_2$.

Therefore, we combine both flipping and cropping to perform data augmentation while preserving the Bayer pattern of the image. After flipping an image, we apply cropping to reverse the change of Bayer pattern. We illustrate



(a) Horizontal Flip (b) Vertical Flip

(c) Bayer Pattern Preserving Augmentation

Figure 4. An example of Bayer preserving augmentation. Since flipping an raw image affects its Bayer pattern, to obtain a horizontal flipped BGGR image from a BGGR image, we first perform a horizontal flipping (BGGR→GBRG), and then crop its first and last column (GBRG→BGGR).

this process in Fig. 4 (c).

As another type of flipping, a transposition has different effects on different patterns, depending on the channels of the diagonal components. Generally, the transpose of a $C_1C_2C_3C_4$ image would be in the pattern of $C_1C_3C_2C_4$. For a RGrGbB input, its transpose would be in RGbGrB, which is roughly the same format (assuming the different between Gr channel and Gb channel is subtle). However, for a GRBG input, its transpose would be in GBRG, a totally different pattern. For this reason, we can safely perform transposition to augment RGGB and BGGR images, but not in GRBG or GBRG.

Training with patches instead of the entire images is another common trick used in model training. Different from the cropping operations in BayerUnify, to correctly obtain patches from the entire Bayer raw image without changing its Bayer pattern, we need to avoid any offset. This could be done by simply cropping even numbers of rows (columns).

With combinations of the discussed three flipping methods and one cropping method, we are able to perform data augmentation on Bayer raw images without any flaw. Note that they can be applied on both homogeneous datasets [8] and heterogeneous datasets [1], enhancing the generalizability of the obtained model.

4. Experiment

4.1. Dataset

We evaluate our method on the Smartphone Image Denoising Dataset (SIDD) [1]. Its training set contains 320 pairs of noise-free images and noisy images, which cover 3 different Bayer patterns and 10 different scenes. Its validation set and testing set consist of 40 pairs of image from 8 different scenes. Both raw images and sRGB images are available.

Set	Scene	# GRBG	# BGGR	# RGGB	Total
Train	2-10	40	126	98	264
Test	1	60	32	20	56

Table 1. Our train/test split of SIDD dataset.

To compare the generalizability of the obtained models, we need to avoid testing on a trained scene. Therefore, we divided the original SIDD training set into two parts: all "Scene 1" image pairs as our testing partition, and the remaining pairs as our training partition. The details are shown in Table 1.

4.2. Network Architecture and Training Details

Table 2 shows the modified U-Net [25] architecture used in our experiments. As proposed by [8], we packed the raw images into 4 channels as the network input. Differently, we trained the networks to produce 4-channel outputs, and unpacked them to obtain denoised raw images.

In our experiments, all the networks were trained with L_1 loss and AdamW optimizer [21] with initial learning rate of $2e-4$ and weight decay of $2e-5$. Patch size and mini-batch size were set to 512 and 4 respectively. We trained each model for $200,000$ iterations, and divided the learning rate by 10 on plateaus. We detected the plateaus and selected the best models using the PSNR scores on the scene 1 patches of the official validation set. For testing, we fed the entire images to the network, and measured the average PSNR scores of the outputs.

4.3. Results and Analysis

To show the effectiveness of our proposed BayerUnify and BayerAug, we compared them with naive training methods. The results of the obtained models are shown in Table 3.

GRBG/BGGR/RGGB Only. As our baseline, we trained one network for each Bayer pattern. Since the number of samples available for training each model is insufficient, this method resulted in a limited performance.

BayerUnify. Another network was trained with our proposed Bayer pattern unification (Sec. 3.1). In the training phase, we applied cropping to convert all 264 training pairs to BGGR format. In the testing phase, we employ padding to unify and cropping to disunify the test cases. Thanks to the increase of training samples, our method outperforms the previous baseline.

BayerUnify+BayerAug. We further trained a network with both Bayer pattern unification and Bayer preserving

Name	# Out	Type
Input	4	Input Image
EncConv1_1	32	Conv 3×3
EncConv1_2	32	Conv 3×3
Pool1	32	Maxpool 2×2
EncConv2_1	64	Conv 3×3
EncConv2_2	64	Conv 3×3
Pool2	64	Maxpool 2×2
EncConv3_1	128	Conv 3×3
EncConv3_2	128	Conv 3×3
Pool3	128	Maxpool 2×2
EncConv4_1	256	Conv 3×3
EncConv4_2	256	Conv 3×3
Pool4	256	Maxpool 2×2
EncConv5_1	512	Conv 3×3
EncConv5_2	512	Conv 3×3
Deconv4	256	Deconv 3×3
Add4	256	Deconv4 + EncConv4_2
DecConv4_1	256	Conv 3×3
DecConv4_2	256	Conv 3×3
Deconv3	128	Deconv 3×3
Add3	128	Deconv3 + EncConv3_2
DecConv3_1	128	Conv 3×3
DecConv3_2	128	Conv 3×3
Deconv2	64	Deconv 3×3
Add2	64	Deconv2 + EncConv2_2
DecConv2_1	64	Conv 3×3
DecConv2_2	64	Conv 3×3
Deconv1	32	Deconv 3×3
Add1	32	Deconv1 + EncConv1_2
DecConv1_1	32	Conv 3×3
DecConv1_2	32	Conv 3×3
DecConv1_3	4	Conv 3×3
Add0	4	Input + DecConv1_3

Table 2. Network architecture used in our experiments. The network takes packed 4-channel raw images as input. Prior to each convolution except EncConv1_1, a PReLU [15] is applied as a pre-activation.

Method	GRBG	BGGR	RGGB
GRBG Only	43.46	-	-
BGGR Only	-	49.50	-
RGGB Only	-	-	51.59
BayerUnify	43.92	49.88	51.85
BayerUnify+BayerAug	**44.02**	**49.92**	**51.95**

Table 3. PSNR of different methods. As shown, our proposed BayerUnify and BayerAug improve the performance on unseen testing samples.

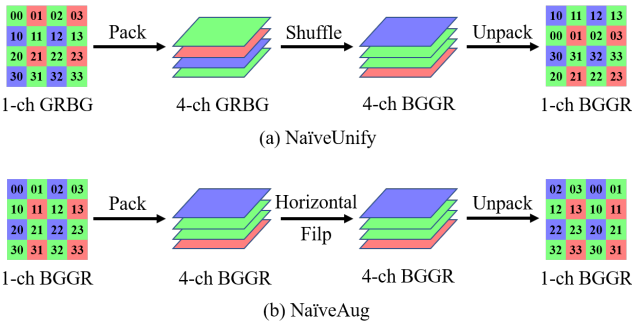


Figure 5. Demonstration of common errors in raw image pre-processing and augmentation. Modifying raw images improperly may disarrange the spatial relationship of the pixels.

Method	GRBG	BGGR	RGGB
NaïveUnify	42.78	49.74	51.83
BayerUnify	**43.92**	**49.88**	**51.85**
BayerUnify+NaïveAug	43.83	49.76	51.81
BayerUnify+BayerAug	**44.02**	**49.92**	**51.95**

Table 4. PSNR of different pre-processing and augmentation methods. As shown, pre-processing and augmenting raw images via problematic methods (NaïveUnify and NaïveAug) result in degradation of the network performance.

augmentation (Sec. 3.2). In the training phase, after unification, we augmented the data via flipping and cropping. The result shows that our data augmentation boosted the generalizability of the obtained model, and consequently improved its performance on the unseen scene.

Since data pre-processing and augmentation on raw images are not as straightforward as they are on RGB images, it is easy to introduce errors when handling them. Next, we discuss some common errors that we mention in Fig. 1, and show how they dampen the performance. The details of these inappropriate modifications are illustrated in Fig. 5, and the results of them are listed in Table 4.

NaïveUnify. A common error in raw image pre-processing is to permute the order of the packed 4-channel input. Demonstrated in Fig. 5 (a), while converting the format, it disorganizes the structure information in the original image. To evaluate, we ran a model with training and testing data unified (to BGGR) with this method. Compared to BayerUnify, this method obtains a lower performance, which shows the importance of our valid raw data pre-processing method.

NaïveAug. In raw data augmentation, it is plausible to flip the packed 4-channel images as we do to 3-channel RGB images. However, as shown in Fig. 5 (b), it also disar-

rays the spatial signal and generates images that are very wandered from the original dataset. We validated this augmentation method based on the correctly unified dataset (BayerUnify). As shown in Table 4, this method dampens the results instead of improving them.

4.4. Challenge Submission

After validating our proposed BayerUnify and BayerAug with a specific train-test separation and a relatively small network, we present our solution in NTIRE 2019 Real Image Denoising Challenge (Track 1) [2]. We applied our proposed methods together with an enhanced network architecture and a model ensembling strategy. Based on the network we mentioned in Sec. 4.2, we increased the network complexity by enlarging the width of earlier stages, and replacing vanilla convolutional layers with residual blocks [16] (with PReLU, without BN [18]). The network architecture is shown in Table 5. In the training stage, we split the 320 training pairs into different categories and trained a model for each one, with both BayerUnify and BayerAug. In the testing stage, we performed BayerUnify to normalize the input patches into BGGR images, classified them according to their metadata, and inferred them using the corresponding models. Our proposed solution achieved a PSNR of 52.11 and a SSIM of 0.9969 on the official test set. We show our results in validation set in Fig. 6.

5. Conclusion

We have presented effective data pre-processing and augmentation methods specifically designed for Bayer raw images, namely BayerUnify and BayerAug. Our results show that the proposed methods ensembled with advanced network architecture can achieve state-of-the-art performance in raw image denoising problem. We also believe raw image processing with deep learning techniques is a promising direction.

Name	# Out	Type
Input	4	Input Image
EncBlock1	256	ResBlock
Pool1	256	Maxpool 2×2
EncBlock2	512	ResBlock
Pool2	512	Maxpool 2×2
EncBlock3	512	ResBlock
Pool3	512	Maxpool 2×2
EncBlock4	512	ResBlock
Pool4	512	Maxpool 2×2
EncBlock5	512	ResBlock
Deconv4	512	Deconv4 3×3
Add4	512	Deconv4 + EncBlock4
DecBlock4	512	ResBlock
Deconv3	512	Deconv 3×3
Add3	512	Deconv3 + EncBlock3
DecBlock3	512	ResBlock
Deconv2	512	Deconv 3×3
Add2	512	Deconv2 + EncBlock2
DecBlock2	512	ResBlock
Deconv1	256	Deconv 3×3
Add1	256	Deconv1 + EncBlock1
DecBlock1	256	ResBlock
PReLU	256	PReLU
DecConv	4	Conv 3×3
Add0	4	Input + DecConv

Table 5. Network architecture used in our NTIRE 2019 Real Image Denoising Challenge (Track 1) submission. Compared to Table 2, we widened the early stages of the network, and replaced convolutional layers with residual blocks.

References

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. A high-quality denoising dataset for smartphone cameras. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[2] Abdelrahman Abdelhamed, Radu Timofte, Michael S. Brown, et al. Ntire 2019 challenge on real image denoising: Methods and results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[3] Michal Aharon, Michael Elad, Alfred Bruckstein, et al. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311, 2006.

[4] Josue Anaya and Adrian Barbu. Renoir–a dataset for real low-light image noise reduction. *Journal of Visual Communication and Image Representation*, 51:144–154, 2018.

[5] Bryce E Bayer. Color imaging array, July 20 1976. US Patent 3,971,065.

[6] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005.

[7] Harold C Burger, Christian J Schuler, and Stefan Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *2012 IEEE conference on computer vision and pattern recognition*, pages 2392–2399. IEEE, 2012.

[8] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3291–3300, 2018.

[9] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1256–1272, 2017.

[10] K Dabov, A Foi, V Katkovnik, and K Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. image processing, ieee transactions on 16 (8), pp. 2080-2095. 2007.

Figure 6. Our results on NTIRE 2019 Real Image Denoising Challenge (Track 1) validation set.

[11] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745, 2006.

[12] Michaël Gharbi, Gaurav Chaurasia, Sylvain Paris, and Frédo Durand. Deep joint demosaicking and denoising. *ACM Transactions on Graphics (TOG)*, 35(6):191, 2016.

[13] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2862–2869, 2014.

[14] Samuel W Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Transactions on Graphics (TOG)*, 35(6):192, 2016.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[17] Keigo Hirakawa and Thomas W Parks. Joint demosaicing and denoising. *IEEE Transactions on Image Processing*, 15(8):2146–2157, 2006.

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[19] Viren Jain and Sebastian Seung. Natural image denoising with convolutional networks. In *Advances in neural information processing systems*, pages 769–776, 2009.

[20] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data. *arXiv preprint arXiv:1803.04189*, 2018.

[21] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 2017.

[22] Julien Mairal, Francis R Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. In *ICCV*, volume 29, pages 54–62. Citeseer, 2009.

[23] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Advances in neural information processing systems*, pages 2802–2810, 2016.

[24] Javier Portilla, Vasily Strela, Martin J Wainwright, and Eero P Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Trans Image Processing*, 12(11), 2003.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[26] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *Proceedings of the IEEE international conference on computer vision*, pages 4539–4547, 2017.

[27] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454, 2018.

[28] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pages 341–349, 2012.

[29] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[30] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

[31] Yuqian Zhou, Jianbo Jiao, Haibin Huang, Yang Wang, Jue Wang, Honghui Shi, and Thomas Huang. When awgn-based denoiser meets real noises. *arXiv preprint arXiv:1904.03485*, 2019.