IEEE Open Journal of Signal Processing

Received 29 January 2024; revised 4 April 2024; accepted 8 April 2024. Date of publication 30 April 2024; date of current version 15 May 2024. The review of this article was arranged by Associate Editor Sergio Benini.

Digital Object Identifier 10.1109/OJSP.2024.3395179

SDAT: Sub-Dataset Alternation Training for Improved Image Demosaicing

YUVAL BECKER, RAZ Z. NOSSEK, AND TOMER PELEG

Samsung Israel R&D Center, SIRC, Tel Aviv 6492103, Israel

CORRESPONDING AUTHOR: YUVAL BECKER (e-mail: yuval.becker@samsung.com).

Code is available at https://github.com/YuvalBecker/SDAT

ABSTRACT Image demosaicing is an important step in the image processing pipeline for digital cameras. In data centric approaches, such as deep learning, the distribution of the dataset used for training can impose a bias on the networks' outcome. For example, in natural images most patches are smooth, and high-content patches are much rarer. This can lead to a bias in the performance of demosaicing algorithms. Most deep learning approaches address this challenge by utilizing specific losses or designing special network architectures. We propose a novel approach **SDAT**, Sub-Dataset Alternation Training, that tackles the problem from a training protocol perspective. SDAT is comprised of two essential phases. In the initial phase, we employ a method to create sub-datasets from the entire dataset, each inducing a distinct bias. The subsequent phase involves an alternating training process, which uses the derived sub-datasets in addition to training also on the entire dataset. SDAT can be applied regardless of the chosen architecture as demonstrated by various experiments we conducted for the demosaicing task. The experiments are performed across a range of architecture sizes and types, namely CNNs and transformers. We show improved performance in all cases. We are also able to achieve state-of-the-art results on three highly popular image demosaicing benchmarks.

INDEX TERMS Demosaicing, image restoration, inductive bias.

I. INTRODUCTION

The task of image demosaicing is a crucial step in digital photography and refers to the process of reconstructing a full-resolution RGB color image from incomplete data obtained from the use of a color filter array (CFA), such as the Bayer pattern of GRBG. In digital cameras, CFA samples only a fraction of the image information, which makes the task of demosaicing complex and challenging. This task is part of a larger group of image restoration problems, such as denoising, deblurring, single image super-resolution, inpainting, removing JPEG compression, etc. These tasks aim at recovering a high-quality image from degraded input data. Image restoration problems are usually considered ill-posed, in the sense that it is usually not possible to determine a unique solution that can accurately reconstruct the original image. This is particularly true for image demosaicing due to the fact that the red, green, and blue color channels are sampled at different locations and rates, which can cause severe aliasing issues. In recent years, the use of convolutional neural networks (CNNs) and transformers has shown significant success in addressing image restoration problems [1], [2], [4], [5], [6] in general, and for image demosaicing in particular, [1], [7], [8], [9], [10], [11], [12]. It is well known that inductive bias, which refers to the assumptions and prior knowledge that the model brings to the learning process, is necessary for the model convergence [13]. But, in some cases, it can have the opposite effect and negatively impact the model's ability to generalize. This is true for deep neural networks as well and has triggered an extensive line of study, e.g. [14], [15], just to mention a few. The bias can arise from various factors such as the network architecture, the training dataset, and others. Our focus in this study is on the inductive bias of a given model caused by the trained dataset. In the case of image demosaicing, biased solutions can cause artifacts, such as zippers, Moiré, color distortion, and more. The most common cause of inductive bias is the well observed phenomenon that natural images are mostly smooth. [16], [17]. However, there are more elements in the data that can cause a given model to induce bias which are less obvious and are more difficult to characterize.

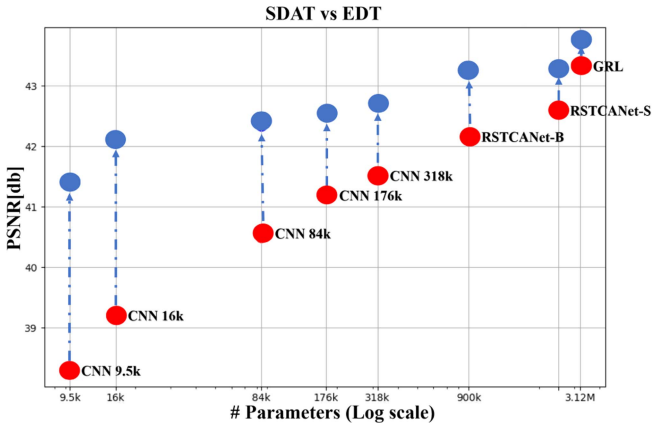

FIGURE 1. A comparison of PSNR results between our training method (SDAT, blue circles) and a standard training on the entire dataset (EDT, red circles) applied on in-house and various popular architectures [1], [2] over Kodak [3] dataset. By using our training method we achieved better results compared to standard training across all architectures. In addition, we achieved state-of-the-art (SOTA) results using GRL architecture.

In this paper we propose **SDAT**, Sub-Dataset Alternation Training, a novel training approach designed to guide convergence towards less biased solutions. Our method extracts sub-datasets from the entire training data, each inducing a unique bias over the trained model. Our training utilizes these sub-datasets to steer the learning process in a non-traditional manner, and prevents converging towards highly likely biased solutions. This approach involves an alternation between training on the entire dataset and training on the extracted sub-datasets.

We demonstrate how SDAT is able to improve the performance of different architectures across all evaluated benchmarks in comparison to the originally implemented training, as can be seen in Fig. 1. One type of evaluation is following a recent trend of low-capacity models (less than 200 k parameters) for edge devices performing image demosaicing [18], [19], [20]. We show that our method is able to effectively utilize the model's capacity and surpass recent relevant works across all benchmarks using fewer number of parameters, showcasing a more efficient solution for low-capacity models. Another is demonstrating improved performance over high capacity models (in an order of 1 M parameters and higher). We evaluate our method both on CNN based architectures and transformers. Thus we exemplify the effectiveness of the proposed method regardless of the architecture. Using SDAT we achieved state-of-the-art on three popular benchmarks by training a variant of the GRL architecture [2].

We summarize our main contributions as follows:

1) We introduce a novel training method for image demosaicing that is able to explore the parameter space more effectively than standard training methods, thus decreasing bias induced by the data.
2) We evaluate our training scheme over various model sizes and different types of architectures, showing great improvement all around and achieving SOTA results over three highly popular benchmarks for image demosaicing.

A comparison of visual results can be found in Fig. 2.

II. RELATED WORK

Image demosaicing aims to restore missing information from the under-sampled CFA of the camera sensor, as in most cases each pixel only captures a single color. This is a well studied problem with numerous methods suggested for solving it. The majority of methods focus on the well known Bayer pattern, which captures only one of the following colors: red, green, or blue at each pixel.

Deep learning methods have become ever more popular, and there has been an active line of study dedicated to performing image demosaicing alone, e.g. [1], [21], [22], [23], [24]. In other cases, it is joined with other tasks (such as image denoising), e.g. [9], [11], [25], [26]. Most of the joint methods still train a network and analyze its performance for the image demosaicing task alone. In [7] the authors designed a joint demosaicing and denoising network using CNNs and constructed a dataset containing solely challenging patches they mined from online data, based on demosaic artifacts. They then went on to train and test the network on cases without the presence of noise. Most works did not follow this practice and trained the demosaicing network on a general dataset (i.e. without applying any mining).

Dedicated task architecture: A prominent line of work focused on architectural modification according to the prior knowledge of the task. Tan et al. [22] proposed a CNN model that first uses bi-linear interpolation for generating the initial image and then throws away the input mosaic image. Given the initial image as the input, the model has two stages for demosaicing. The first stage estimates green and red/blue channels separately while the second stage estimates three channels jointly. The idea of reconstructing the green channel separately or reconstructing it first to use later on as a guidance map is also used by several others, such as Cui et al. [27], and Liu et al. [9]. Other works compared the effect of convolution kernels of different sizes on the reconstruction, e.g. the work by Syu et al. [21], which concluded the larger the size of the convolution kernel, the higher the reconstruction accuracy. Inspired by [21], Gou et al. [28], suggested adapting the Inception block of [29] to reduce computation while still having a large receptive field.

General architectural modification: Another recent popular line of study is utilizing a general neural network architecture and training it separately on various image restoration tasks (i.e. each task requires a separate model with different weights), such as [4], [5], [6], [30], [31]. Zhang et al. suggested [5] the RNAN architecture, which is a CNN network. Xing and Egiazarian [1] suggested slight modifications to the SwinIR architecture of [4] for the specific task of demosaicing. PANet [6] performed a multi-scale pyramid attention mechanism. Li et al. [2] proposed the GRL transformer-based architecture for various image-to-image restoration tasks. Specifically for the demosaicing task, their suggested method

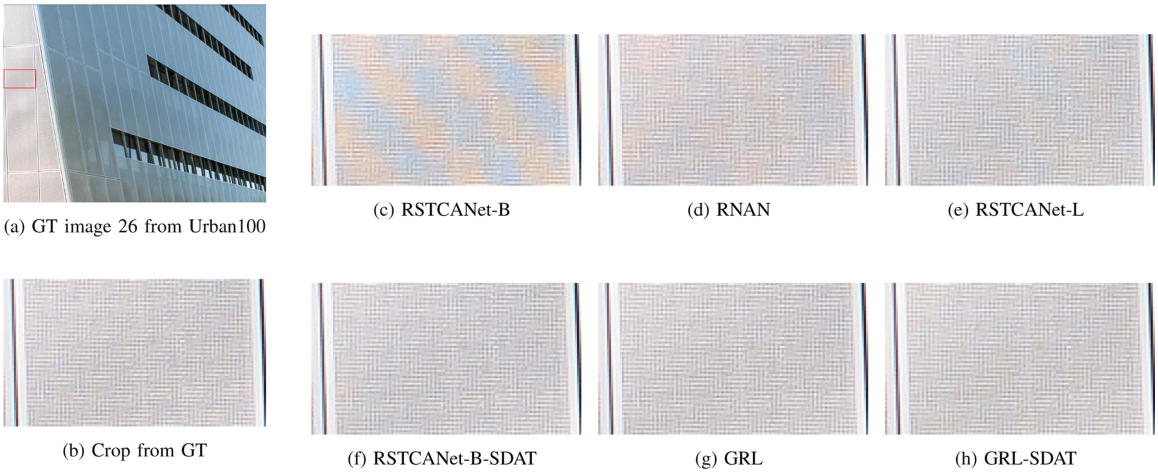

FIGURE 2. Qualitative comparison of our method compared to other top methods: RNAN, RSTCANet, and GRL. The RNAN model has 9M parameters, RSTCANet presents three different model sizes: B, S, and L, having 0.9M, 3.1M, and 7.1M parameters respectively and GRL consists of 3.1M parameters. We demonstrate the results of our suggested training scheme to train the RSTCANet-B and GRL models. The RSTCANet-B-SDAT model produces superior qualitative results compared to all original RSTCANet variants while having the least amount of parameters.

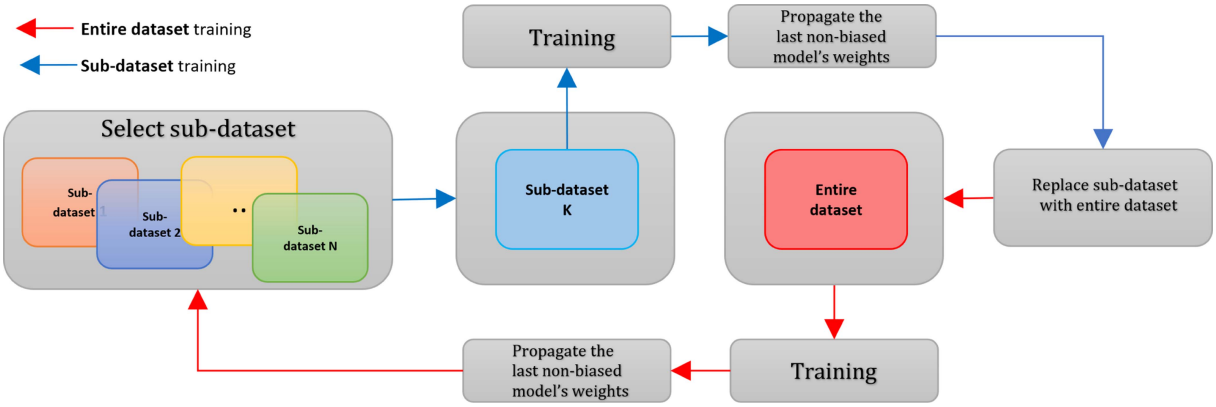

FIGURE 3. An illustration of the SDAT method. Each cycle consists of two training phases. The first consists of training over a specific sub-dataset obtained from the pool of collected sub-datasets as explained in section B, while the second, consists of training over the entire dataset. Each phase is initialized by the model's weights that achieved the lowest validation loss across all sub-datasets in the previous phase. Every cycle a different sub-dataset is selected. The number of cycles depends on the number of categories and architecture.

achieved SOTA accuracy over KODAK [3], McMaster [32], Urban [33] and CBSD68 [34] datasets. Their solution is focused on artifacts removal, as their network receives an initial demosaic solution.

Low-capacity demosaic solutions: It is noteworthy to mention another line of work that focuses on the implementation of low-capacity networks for image demosaicing. In [19] Ramakrishnan et al. employed Neural Architecture Search (NAS) and optimization techniques, while in [20] Wang et al. proposed a custom UNet architecture. Both approaches evaluate their solutions across different model sizes.

Our method (see Fig. 3 for an overview) exhibits several significant distinctions from prior research, as most of the works focused on custom losses and architectural modifications [1], [4], [5], [24]. This work focuses on convergence and training perspectives. Unlike mining-based methods [7] that rely on error criteria to identify only hard examples, our approach employs different metrics and uses a novel identification mechanism to identify multiple sub-datasets, that exhibit different characteristics, each inducing a different bias into the trained model. Furthermore, our training method utilizes the sub-datasets to steer and guide convergence while training on the entire dataset.

III. METHOD

In the following, we present the required stages to perform SDAT. We will show that by applying our training regime we are able to better utilize the given dataset, i.e. achieve better results for a given network architecture without modification to it or the loss function. The method involves training alternatively between the entire dataset and the various sub-datasets in a cyclic fashion. Each of the sub-datasets induces a distinct

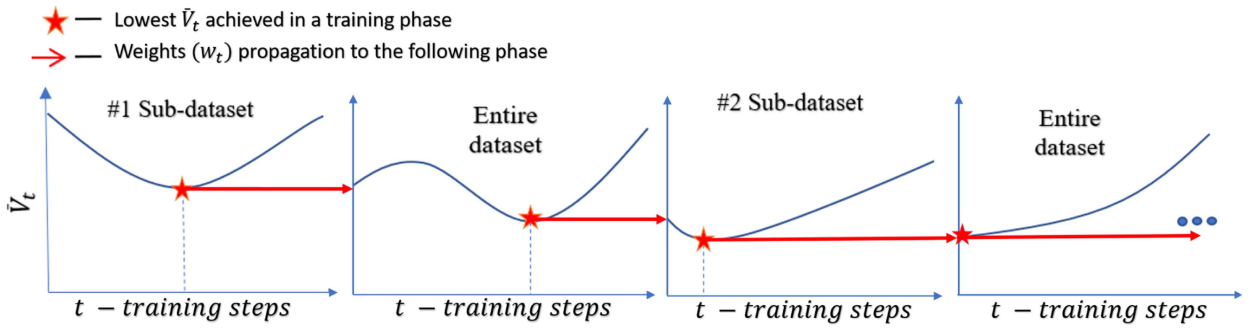

FIGURE 4. Depiction of the weight propagation process of SDAT. An illustration of the alternation process between a sub-dataset and the entire dataset. Each graph is a training phase over a single dataset, where the horizontal axis is the training steps and the vertical axis is the average validation loss across all sub-datasets marked with $\bar{V}_t$. The star on each of the graphs marks the iteration index each training phase stopped accumulating gradients for a specific dataset, as it propagates model weights achieved at that index to the following training phase. Furthermore, there can be training phases that do not aggregate any gradients. As can be seen on the rightmost graph the model could not achieve a convergence that lowers $\bar{V}_t$, therefore, it accumulated zero training steps.

bias over the trained model. SDAT consists of several traversals, where each traversal means, a complete training cycle over all sub-datasets and the entire dataset.

We first outline the details of our suggested training method and explain how it aids in steering the convergence towards a less biased solution by exploiting the specific characteristics of the collected sub-datasets. Then we provide a detailed explanation of our process for gathering these sub-datasets and how we generated them to possess their distinct characteristics as mentioned above.

A. TRAINING METHOD

Our training method as depicted in Fig. 3 incorporates two primary mechanisms:

1) *Dataset alternation:* The training process consists of alternating between different training phases. We alternate between a collected sub-dataset and the entire dataset. This alternation takes place every predetermined number of iterations. At each cycle, a different sub-dataset is used.
2) *Solution selection process:* During each training phase we monitor every iteration the average of all sub-datasets' validation loss along with the corresponding model's weights:

$$\bar{V}_t = \frac{1}{N}\sum_{i=1}^{N} V_{w_t,c_i} \tag{1}$$

where t is the iteration number (training step), V_{w_t,c_i} is the average validation value of sub-dataset c_i, given model's weights w_t, and N is the number of sub-datasets.

The chosen model's weights for the next alteration are the w_t obtained the lowest $\bar{V}_t$ during the training phase. See Fig. 4 for a more detailed example. We observed that a model consistently achieving low validation loss across all sub-datasets is more likely to converge towards a non-biased solution.

These two key factors combined help in guiding the convergence towards a less biased solution. Training over the entire dataset limits the exploration of the solution space due to the inherent bias of the dataset. By training over the sub-datasets, we are able to strongly steer the convergence towards a wider range of solutions, as each sub-dataset induces a different bias on the convergence.

Moreover, we maintain the usage of the entire dataset, as it contains the overall statistics of the task and every second alternation we train over the entire dataset. We are able to achieve a solution that does not converge to a biased local minimum due to the selection process, as it only propagates to the following training phase the least biased solution achieved during the current training phase. This way we are able to enjoy both worlds - we can still use the entire dataset for training, without being limited only to the solutions governed by the inherent dataset bias.

Sub-datasets convergence rate: When training on the entire dataset, simple functions are converged first and complex functions are converged later as demonstrated in prior research [35], [36]. Our models also exhibited a similar trend. The initial convergence was governed by the entire dataset bias (in our case due to smooth patches in natural images), hindering further convergence toward more complex elements (high frequency, edges and more). In every training phase over a specific sub-dataset, the accumulation of gradients is till the last non-biased update as in the following:

$$\Gamma_{c_i} = \sum_{t=t_{ci}}^{\hat{t}_{ci}} \eta_t \frac{d}{d w_t} \mathcal{L}_{w_t,c_i}$$
$$\hat{t}_{ci} = \underset{t}{\operatorname{argmin}}\, \bar{V}_t \tag{2}$$

where Γ_{c_i} denotes the accumulated gradients obtained during a training session of a specific sub-dataset, t_{ci} and $\hat{t}_{ci}$ denote the training starting and ending time index respectively, using sub-dataset c_i, η_t denotes the learning rate value in each time

step and $\mathcal{L}_{w_t,c_i}$ denotes the training loss function given the model's weights w_t and training over sub-dataset c_i.

After each traversal over all sub-datasets, the model's weights are composed based on the accumulated custom gradients from each sub-dataset:

$$w_{traversal} = w_0 + \sum_{i=1}^{i=N} \Gamma_{c_i} + \Gamma_{c_0} \tag{3}$$

where w_0 is the weights of the model at the beginning of the traversal, Γ_{c_0} is the accumulated gradients obtained along the entire dataset training phases.

As demonstrated in [35], [36] and observed empirically in our experiments, controlling the convergence rate over different elements in the dataset leads to different solutions. Our training mechanism adapts the convergence rate of each sub-dataset as in (3), in order to obtain the least biased solution.

Learning rate scheduler: To ensure stability during training, every dataset alternation we use a learning rate (LR) scheduler that starts with a low value and gradually increases to a higher value of LR. Shifting between datasets significantly impacts the gradients at the current weight space position, altering the structure of the loss weight landscape and requiring readjustment of the LR. Only the weights that achieved the lowest validation loss across all sub-datasets in each training session are carried over to the next alternation as in (1) and (2), filtering out unstable solutions that might arise from unsuitable LR.

B. SUB-DATASETS IDENTIFICATION AND CREATION

We now describe the collection of sub-datasets from the entire dataset, each inducing a different bias on the trained model. This phase consists of the following:

1) *Generation of sub-datasets:* To establish a starting point for data categorization, we performed a training using the entire dataset to obtain a base model. We used a combination of existing metrics, such as L_1, perceptual [37], and additional custom metrics, (5) to (7), to detect a wide variety of artifact types created by the base model's outputs. we denote the initial predicted patch $\hat{P}_{init}$ and the patch in the ground-truth RGB image as P, both of size $H_p \times W_p \times 3$.
   a) *Zipper metric* - this metric d_{zip} is used to find zipper artifacts near edges in the image. First, we calculate a mask M_{NE} to identify non-edge areas as follows

$$M_{NE} := |C(\hat{P}_{init}) - C(P)| < \varepsilon_1, \tag{4}$$

   where C is some kind of edge detector, *e.g.* Canny, and ε_1 is a predefined threshold. Then we are able to calculate

$$d_{zip} := \| \, |\nabla \hat{P}_{init}| - |\nabla P| \, | \odot M_{NE} \|_1, \tag{5}$$

   b) *Grid metric* - this metric d_{grid} is used to find grid-like artifacts in flat areas as follows

$$d_{grid} := \| 2 \cdot |\nabla \hat{P}_{init}| \odot \left(\sigma \left(\frac{\alpha}{|\nabla P|} \right) - 0.5 \right) \odot (|\nabla \hat{P}_{init}| > \varepsilon_2) \|_1, \tag{6}$$

   where σ is the Sigmoid function, $\alpha > 0$ is a parameter that scales the values of the Sigmoid, and ε_2 is another predefined threshold.
   c) *Edge distance* - This metric d_{edge} is used to find distortions around edges

$$d_{edge} := \| C(\hat{P}_{init}) - C(P) \|_1. \tag{7}$$

We note that by using different thresholds in (4) and (6) we create a diverse pool of sub-datasets within the data. Examples of some of the identified patch categories are shown in Fig. 5.

a) *Selection and elimination of sub-datasets:* From the collected pool of sub-datasets we keep the sub-datasets, that training on them induces a bias that hinders the generalization of the model on the entire dataset. These sub-datasets induce a different bias than the entire dataset and eventually help converge to a better solution. This procedure is illustrated in Fig. 6 and is composed of the following two steps:
   a) *Searching for inverse-correlation:* We conduct brief training sessions overall sub-datasets generated in the previous stage. Each session trains over a single sub-dataset and starts from the model that was trained over the entire dataset, as described in the previous stage. Throughout each session, we evaluate the validation loss over each of the different sub-datasets, including the entire dataset. We search for a negative correlation, in the sense of Spearman's rank correlation coefficient, between the validation loss on the trained sub-dataset and the validation loss on the entire dataset. If indeed there is a negative correlation, we identify a sub-dataset that the model struggles to generalize using the bias induced by the entire dataset. This means the model must discard its data-induced bias to achieve convergence for that specific sub-dataset.
   b) *Merging sub-datasets:* When we detect sub-datasets with a positive correlation response between their validation losses (in the same sense as in step (a)), it means that training over one of the sub-datasets does not harm the model's generalization over the other sub-dataset. Therefore, these sub-datasets are merged into a single sub-dataset.

In summary, we identify certain sub-datasets that are susceptible to bias, where each of the final generated sub-datasets is likely to induce a unique bias over the trained model. Since a model that has converged toward a biased solution is likely to produce a high validation loss (see (1)) over these sub-datasets, it is used as a metric to estimate a biased solution.

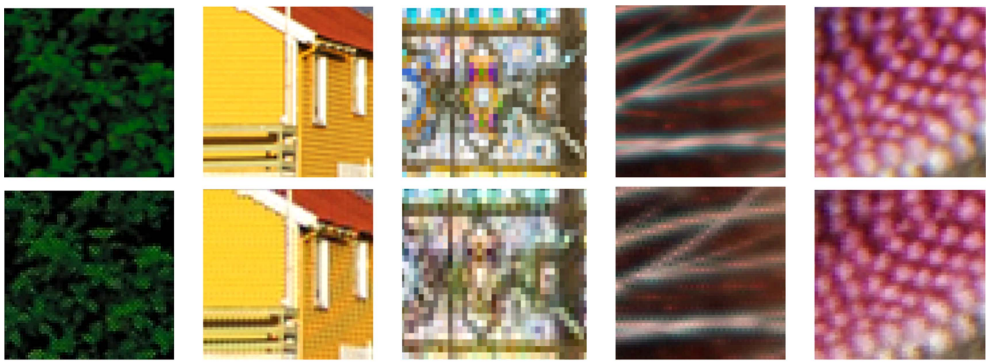

FIGURE 5. Examples of sub-datasets patches that are found via the different metrics. Each metric focuses on different aspects of image demosaicing. The top row depicts the ground truth patches, and the bottom is the prediction of the network after vanilla training. (a) A patch with grid artifacts, (b) a patch with zipper artifact, (c) a patch with high $L1$ distance, (d) a patch with high edge distance, and (e) a patch with high perceptual distance.

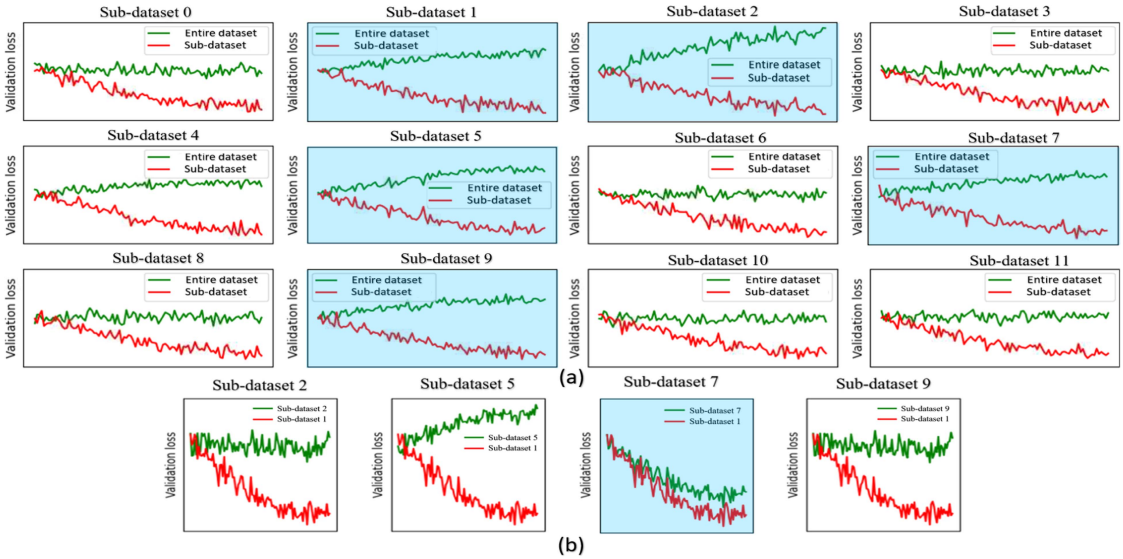

FIGURE 6. Our two step elimination process for selecting the sub-datasets. (a) shows the first step, where each graph represents a training session over a sub-dataset (the horizontal axis shows the number of epochs, and the vertical axis shows the validation error). The green line represents the validation error of the entire dataset and the red line represents the validation error on the trained sub-dataset. Validation errors of sub-datasets with strong negative correlation with the entire dataset validation error, are selected and highlighted in light blue. (b) depicts an example of the second step. The validation error of each sub-dataset [from those chosen in (a)] is correlated in the same sense in step (a) with the validation error of the other sub-datasets. The objective is to merge the sub-datasets that demonstrate positive correlation, resulting in a similar bias induced by the trained model. In the example, we demonstrate sub-dataset 1's validation curve (red line) with the other chosen sub-datasets (green line) from the previous step. The sub-datasets chosen to be merged with sub-dataset 1 are highlighted in light blue.

IV. EXPERIMENTS

In order to evaluate our suggested training method we divided our experiments into two architecture groups that might be affected differently: (1) Low-capacity models and (2) High-capacity models. Over each architecture, we conducted two training types: the first is a training over the entire dataset, which we refer to as EDT (Entire Dataset Training) and the second is our proposed SDAT method.

For both types of training settings, we used crops of size 64×64 as the input and a batch size of 16. We also used data augmentations, such as a random flip and transpose over both image axes during training. All models were trained using the L_1 norm between the ground truth and estimated RGB images as the loss function.

EDT method settings: we used the DIV2K [38] dataset for training. We used the Adam optimizer [39] with learning rate 5×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$. Every 200 k iteration we lower the learning rate by a factor of 2. The models were trained for a total of 2 M iterations.

SDAT method settings: In addition to using the DIV2K dataset as a whole, we employed the process described in Section B, and extracted five sub-datasets from the DIV2K

dataset. We used the same experiment parameters as in the EDT, except for the suggested modification to the LR scheduler, as explained in Section A. For every sub-dataset alternation, we initialize the LR to 1×10^{-5}, and it is linearly increased to 1×10^{-3} by the end of the training session over the sub-dataset. Important to clarify, that the optimizer parameters for the EDT phase in each cycle are identical to the parameters presented in the above EDT method settings.

Our training process encompassed 20 complete traversals. Each traversal was designed to cover a full training cycle across all sub-datasets, where a single cycle involved sequential training first on a sub-dataset and then on the entire dataset. For each phase of training within a cycle, we allocated 10 k iterations, totaling 20 K iterations per cycle, 100 K iterations per traversal, and 2 M iterations per the entire training session. Monitoring rate: we monitor $\bar{V}_t$ as in (1), every 1 K iterations.

We chose to train based on a specific number of iterations, rather than traditional epochs, due to the varying sample sizes across datasets.

It should be noted that SDAT takes more time to converge compared to EDT because of its more complex data setup and the solution selection process (section A) that propagates to the following alternation the least bias weights, consequently disregarding subsequent training iterations. However, to ensure a fair comparison, we've chosen to run the same number of training iterations for both methods.

A. LOW-CAPACITY MODELS

We trained three CNN architectures with 9.5 K, 16 K, and 84 K parameters. Our CNN has a similar structure to the overall structure of DemosaicNet suggested by Gharbi et al. in [7], with slight differences. We replaced each of the two convolution layers with 3 Inverted Linear bottlenecks (ILB) [40]. We obtain different architecture sizes by adjusting the expansion size parameters of the ILB. In addition to comparing the results between EDT and our suggested method, we also compared our results (in terms of PSNR) with recent studies that focus on achieving high performance for image demosaicing using low capacity models [19], [20]. As indicated in Table 1, our training method presents superior results across all benchmarks compared to the EDT method. Additionally, while our trained CNNs have the lowest number of parameters in each sub-range of parameters, the networks outperform all other methods across all benchmarks.

B. HIGH-CAPACITY MODELS

We evaluated our SDAT method in comparison to the EDT method over three leading transformer architectures. First, we implemented RSTCANet [1], which is based on the SwinIR [4] architecture. We applied both methods over two RSTCANet variants, RSTCANet-B and RSTCANet-S with 0.9 M and 3.1 M parameters respectively. Additionally, we trained a variant of the GRL-S [2] architecture with 3.1 M parameters. As can be seen in Table 2, we evaluated four popular benchmarks that represent natural images, with additional two benchmarks HDR-VDP and Moiré initially presented in [7], and are known as a collection of hard patches. Our training method produced superior results overall architectures and across all benchmarks in comparison to the EDT method described above. It is worth mentioning, that while the EDT method yielded comparable or even superior results to the published results across all RSTCANet variants, it could not reconstruct the reported results for the GRL. However, as can be seen in Table 3 by using the SDAT method to train the GRL architecture, we achieved SOTA over three popular benchmarks and second best in the fourth.

TABLE 1. PSNR Results for Image Demosaicing for Compact Networks

Parameter range	Method	Params.	Kodak	MCM	Urban100
9K–12K	Ark Lab [19]	10K	40.4	36.9	-
	Wang [20]	11.7K	40.8	36.75	36.4
	CNN EDT	9.5K	38.3	35.4	35.44
	CNN SDAT	9.5K	**41.39**	**37.02**	**36.79**
16K–20K	Ark Lab	20K	40.9	37.6	-
	CNN EDT	16K	39.31	35.81	36.2
	CNN SDAT	16K	**42.05**	**37.80**	**37.52**
80K–200K	Ark Lab	200K	41.2	38.2	-
	Wang	183K	41.72	37.91	37.7
	CNN EDT	84K	40.25	37.44	37.04
	CNN SDAT	84K	**42.38**	**38.52**	**38.01**

The best results are highlighted in bold.

C. ABLATION STUDY

Comparing different training methods: To verify the contribution of our suggested training method we conducted the following ablation study to compare our method's effectiveness versus other training methods: (1) standard training over the entire dataset, (2) standard training over hard mined examples only (similar to [7]), (3) standard training over a uniform distribution, i.e. we compose a new dataset where the probability to sample a training example from the entire dataset and from the sub-datasets is equal, (4) performing our method, but discarding the alternation to train on the entire dataset, and (5) performing SDAT, but using sub-datasets that were randomly sampled from the entire dataset, instead of the sub-datasets created using the suggested method in Section B. The training procedure for the standard training methods (1)–(3) is the same as described for EDT settings above. As indicated in Table 4, our method surpasses all other training regimes, by a considerable margin. Compared to standard training, we conducted a wider experiment covering a wide range of model sizes using the same training procedure described above. The results are shown in Fig. 1. It can be seen how we are able to improve the networks' performance between 1-3.5 dB depending on the initial model size (the smaller the network, the higher the performance boost) and the evaluation dataset.

Comparing different numbers of sub-datasets: As outlined in Section A, our approach involves switching between a

TABLE 2. PSNR Results for Image Demosaicing for Leading Transformer Architectures

Method	MCM	Kodak	CBSD68	Urban100	HDR-VDP	Moiré
RSTCANet-B-EDT	38.83	42.22	41.83	38.45	33.6	36
RSTCANet-B-SDAT	**39.72**	**43.22**	**42.58**	**39.88**	**34.52**	**37.81**
RSTCANet-S-EDT	39.52	42.71	42.44	39.72	34.34	37.53
RSTCANet-S-SDAT	**39.75**	**43.31**	**42.82**	**40.14**	**34.72**	**37.92**
GRL-EDT	39.7	43.14	42.95	40.05	34.7	38.1
GRL-SDAT	**40.11**	**43.65**	**43.31**	**40.67**	**34.95**	**38.3**

We compare standard training over the entire dataset (EDT) with our suggested training approach (SDAT). The best results for each architecture are in bold.

TABLE 3. Benchmark Results (PSNR and SSIM) for Image Demosaicing

Method	Params.	MCM		Kodak		CBSD68		Urban100	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
DRUNet[31]	11M	39.40	0.991	42.30	0.994	42.33	0.995	39.22	0.990
RNAN [5]	9M	39.71	0.972	43.09	0.990	42.50	0.992	39.75	0.984
JDD [25]	3.5M	38.85	0.990	42.23	0.994	-	-	38.34	0.989
PANet [6]	6M	40.00	0.973	43.29	0.990	42.86	0.989	40.50	0.985
RSTCANet-B [1]	0.9M	38.89	0.990	42.11	0.994	41.74	0.9954	38.52	0.990
RSTCANet-S [1]	3.1M	39.58	0.991	42.61	0.995	42.36	0.995	39.69	0.992
RSTCANet-L [1]	7.1M	39.91	**0.991**	42.74	0.995	42.47	0.996	40.07	0.993
GRL [2]	3.1M	40.06	0.988	43.34	0.995	**43.89**	**0.997**	40.52	0.992
RSTCANet-B **SDAT**	0.9M	39.72	0.98	43.22	0.995	42.58	0.996	39.88	0.991
RSTCANet-S **SDAT**	3.1M	39.75	0.990	43.31	0.995	42.82	0.996	40.14	**0.993**
GRL **SDAT**	3.1M	**40.11**	0.988	**43.65**	**0.996**	43.31	0.996	**40.67**	0.992

The best results are in bold, and the second best are underlined. Except for the results obtained by our SDAT method, the rest of the results in this table were obtained by the official code provided by the authors or reported by them in the original papers.

TABLE 4. PSNR Results of Various Training Methods Compared to Ours Over the Kodak and McMaster Datasets

Params.	EDT		Mined training		Uniform distribution training		SDAT without EDT phase		SDAT with random sub-datasets		SDAT	
	Kodak	MCM	Kodak	MCM	Kodak	MCM	Kodak	MCM	Kodak	MCM	Kodak	MCM
16K	39.20	35.90	40.20	36.40	40.10	37.50	39.70	37.50	39.11	35.71	**42.05**	**37.80**
176K	40.55	37.20	40.90	37.50	41.10	38.20	41.00	38.00	40.32	37.11	**42.41**	**38.67**

We evaluated each training method using two architectures consisting of 16K and 176K parameters. The best results are in bold, and the second best are underlined.

TABLE 5. PSNR Results Over Kodak and MCM Datasets Using a Different Number of Sub-Datasets for Training

Params.	1 sub-dataset		2 sub-datasets		5 sub-datasets	
	Kodak	MCM	Kodak	MCM	Kodak	MCM
16K	39.22	36.21	40.50	36.54	**42.05**	**37.80**
176K	40.50	37.25	41.25	37.77	**42.41**	**38.67**

The best results are in bold.

collection of sub-datasets and the entire dataset. In this experiment, we assess the effectiveness of our training method by randomly selecting one and two sub-datasets. The results, as shown in Table 5, demonstrate that the number of sub-datasets included in the alternation process has a significant impact on the resulting performance. This demonstrates that being able to identify additional types of bias in the original dataset can further improve the network's overall generalization.

Different traversal order over the sub-datasets: As the SDAT method alternates among different sub-datasets, we investigated whether the sequence in which these sub-datasets are accessed during a traversal impacts the outcome. We carried out four separate experiments, with each one involving a unique sequence of traversing through the selected five sub-datasets. As can be seen in Fig. 7, there is no significant impact on the performance of the 16 K and 176 K models based on the sequence in which the sub-datasets are traversed. Across all the experiments, the largest standard deviation recorded was 0.05db, not significant enough to suggest any form of dependency. One plausible explanation is that our training protocol shifts between datasets while monitoring every iteration so that we don't converge toward a biased solution as explained in section A using the solution selection process. This mechanism regularizes the convergence rate of all sub-datasets, preventing over-fitting on any particular sub-dataset.

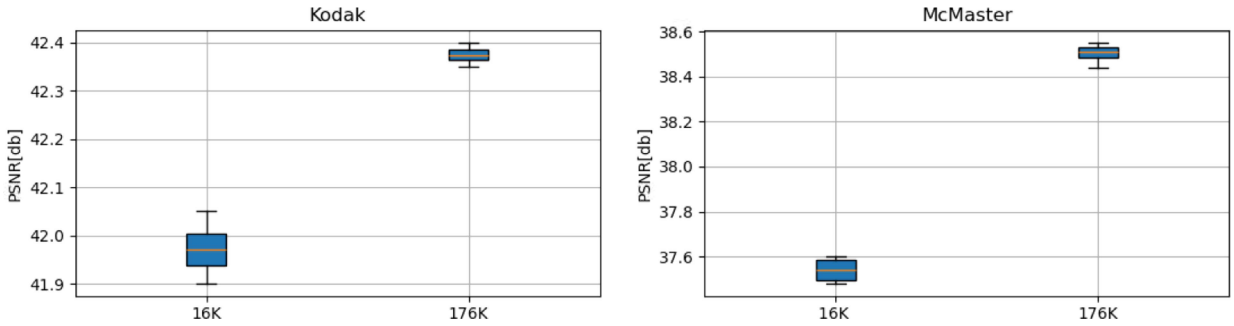

FIGURE 7. Evaluation of the obtained models using SDAT for different sub-dataset traversal sequences. This evaluation involved testing four distinct sequences with both our 16K and 176K models on the Kodak and mcm datasets. To provide a clearer visualization of the outcomes' distribution, we depicted the results through box plots.

It aims to ensure that every sub-dataset is adequately represented, as the criterion penalizes the model's inability to generalize effectively across any given sub-dataset.

Random sub-datasets: To evaluate the importance of the identification phase as in section B, instead of obtaining the most significant sub-datasets according to the predefined criterion, we collected randomly five sub-datasets. As can be seen in Table 4, under "SDAT with random sub-datasets". This method yielded results that were not as effective as those achieved by SDAT, emphasizing the importance of the identification stage. Additionally, it produced results inferior to the EDT, which can be explained due to the difference in training steps between SDAT and EDT. As detailed in the "solution selection process", SDAT effectively conducts fewer weight updates for an equivalent number of training steps compared to EDT, further explained in Section V.

V. DISCUSSION

With SDAT we presented a novel training scheme improving the results (in the sense of PSNR and SSIM) for the image demosacing task across various types of Deep Learning model sizes and architectures. Despite the promising results obtained using SDAT, some limitations should be acknowledged. One such limitation is the complexity involved in generating the sub-datasets, on which we suggest to perform the alternation during training. As outlined in Section B the creation and identification of these sub-datasets requires the use of various metrics. This requires domain-specific expertise. Thus, extending the use of SDAT to other image restoration tasks, such as image denoising, single image super-resolution, and mitigating JPEG compression, might require the use of different metrics. On top of the current method for creating new sub-datasets being time consuming, finding adequate metrics for different tasks may not be straightforward. The creation of the sub-datasets is domain-specific, and is done in a semi-manual process. A potential future direction could involve performing clustering in some latent space, where the features encapsulate meaningful information regarding different types of bias. Such an approach may offer the advantage of not relying on prior knowledge specific to the task being addressed. As a result, it can enable the application of SDAT to a range of different problems, while also facilitating a more automated process. Additionally, as pointed out in Section IV, due to the solution selection process (section A) SDAT training might perform inefficient training steps. A possible improvement can be in the form of an estimator, that assesses at each iteration whether the current sub-dataset training settled into a local minimum, then, stop the current sub-dataset training and move to the subsequent phase, as the following iterations will most likely not achieve better $\bar{V}_t$.

VI. CONCLUSION

In this paper, we addressed the challenge of dataset inductive bias for the image demosaic task and proposed SDAT, a novel training method to improve the performance of a model. The method involves two main steps: defining and identifying sub-datasets beneficial to model convergence and an alternating training scheme. SDAT demonstrated an improved performance for image demosaicing over standard training methods and achieved state-of-the-art results on three highly popular benchmarks. Our suggestion also effectively utilized the model's capacity, surpassing recent relevant works on low-capacity demosaicing models across all benchmarks using fewer parameters. This result is crucial for edge devices. The method's success also demonstrates the importance of considering the dataset structure in optimizing a model's training process. Our findings suggest that our proposed method can be applied to other image restoration tasks and can be used as a trigger for further research in this field.

REFERENCES

[1] W. Xing and K. Egiazarian, "Residual swin transformer channel attention network for image demosaicing," in *Proc. 10th Eur. Workshop Vis. Inf. Process.*, 2022, pp. 1–6.
[2] Y. Li et al., "Efficient and explicit modelling of image hierarchies for image restoration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 18278–18289.
[3] X. Li, B. Gunturk, and L. Zhang, "Image demosaicing: A systematic survey," *Proc. SPIE*, vol. 6822, pp. 489–503, 2008.
[4] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "SwinIR: Image restoration using swin transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 1833–1844.
[5] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, "Residual non-local attention networks for image restoration," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2019, pp. 5517–5534.

[6] Y. Mei et al., "Pyramid attention network for image restoration," *Int. J. Comput. Vis.*, vol. 131, pp. 3207–3225, 2023.
[7] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand, "Deep joint demosaicking and denoising," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–12, 2016.
[8] F. Kokkinos and S. Lefkimmiatis, "Deep image demosaicking using a cascade of convolutional residual denoising networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 303–319.
[9] L. Liu, X. Jia, J. Liu, and Q. Tian, "Joint demosaicing and denoising with self guidance," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2240–2249.
[10] Y. Niu, J. Ouyang, W. Zuo, and F. Wang, "Low cost edge sensing for high quality demosaicking," *IEEE Trans. Image Process.*, vol. 28, no. 5, pp. 2415–2427, May 2019.
[11] G. Qian et al., "Rethinking learning-based demosaicing, denoising, and super-resolution pipeline," in *Proc. IEEE Int. Conf. Comput. Photogr.*, 2022, pp. 1–12.
[12] D. S. Tan, W.-Y. Chen, and K.-L. Hua, "DeepDemosaicking: Adaptive image demosaicking via multiple deep fully convolutional networks," *IEEE Trans. Image Process.*, vol. 27, no. 5, pp. 2408–2419, May 2018.
[13] T. M. Mitchell, "The need for biases in learning generalizations," Citeseer, 1980. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=6cf35ec34efa592f83e3a1b748aea14957fc784a
[14] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness," in *Proc. Int. Conf. Learn. Representations*, 2019.
[15] S. Ritter, D. G. Barrett, A. Santoro, and M. M. Botvinick, "Cognitive psychology for deep neural networks: A shape bias case study," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2940–2949.
[16] A. Hyvärinen, J. Hurri, and P. O. Hoyer, *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision*, vol. 39. London, U.K.: Springer, 2009.
[17] M. Zontak and M. Irani, "Internal statistics of a single natural image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 2011, pp. 977–984.
[18] K. Ma et al., "Searching for fast demosaicking algorithms," *ACM Trans. Graph.*, vol. 41, no. 5, pp. 1–18, 2022.
[19] R. Ramakrishnan, S. Jui, and V. Partovi Nia, "Deep demosaicing for edge implementation," in *Proc. Image Anal. Recognit.: 16th Int. Conf.*, 2019, pp. 275–286.
[20] S. Wang, M. Zhao, R. Dou, S. Yu, L. Liu, and N. Wu, "A compact high-quality image demosaicking neural network for edge-computing devices," *Sensors*, vol. 21, no. 9, 2021, Art. no. 3265.
[21] N.-S. Syu, Y.-S. Chen, and Y.-Y. Chuang, "Learning deep convolutional networks for demosaicing," 2018, *arXiv:1802.03769*.
[22] R. Tan, K. Zhang, W. Zuo, and L. Zhang, "Color image demosaicking via deep residual learning," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2017, vol. 2, no. 4, pp. 793–798.
[23] T. Kerepecky, F. Šroubek, A. Novozámsky, and J. Flusser, "NERD: Neural field-based demosaicking," in *Proc. IEEE Int. Conf. Image Process.*, 2023, pp. 1735–1739.
[24] H. Lee et al., "Efficient unified demosaicing for bayer and non-bayer patterned image sensors," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 12750–12759.
[25] W. Xing and K. Egiazarian, "End-to-end learning for joint image demosaicing, denoising and super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 3507–3516.
[26] T. Zhang, Y. Fu, and C. Li, "Deep spatial adaptive network for real image demosaicing," in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 3, pp. 3326–3334.
[27] K. Cui, Z. Jin, and E. Steinbach, "Color image demosaicking using a 3-stage convolutional neural network structure," in *Proc. 25th IEEE Int. Conf. Image Process.*, 2018, pp. 2177–2181.
[28] Y. Guo, Q. Jin, G. Facciolo, T. Zeng, and J.-M. Morel, "Residual learning for effective joint demosaicing-denoising," 2020, *arXiv:2009.06205*.
[29] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-V4, inception-resnet and the impact of residual connections on learning," in *Proc. AAAI Conf. Artif. Intell.*, 2017. pp. 4278–4284.
[30] C. Mou, J. Zhang, and Z. Wu, "Dynamic attentive graph learning for image restoration," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 4328–4337.
[31] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte, "Plug-and-play image restoration with deep denoiser prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6360–6376, Oct. 2022.
[32] L. Zhang, X. Wu, A. Buades, and X. Li, "Color demosaicking by local directional interpolation and nonlocal adaptive thresholding," *J. Electron. Imag.*, vol. 20, no. 2, 2011, Art. no. 23016.
[33] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5197–5206.
[34] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th IEEE Int. Conf. Comput. Vis.*, 2001, vol. 2, pp. 416–423.
[35] N. Rahaman et al., "On the spectral bias of neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5301–5310.
[36] R. Basri, M. Galun, A. Geifman, D. Jacobs, Y. Kasten, and S. Kritchman, "Frequency bias in neural networks for input of non-uniform density," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 685–694.
[37] C. Ledig et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 105–114.
[38] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 1122–1131.
[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015.
[40] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.