A Two-Stage Convolutional Neural Network for Joint Demosaicking and Super-Resolution

Kan Chang, Hengxin Li, Yufei Tan, Pak Lun Kevin Ding, *Member, IEEE*, and Baoxin Li, *Senior Member, IEEE*

***Abstract*— As two practical and important image processing tasks, color demosaicking (CDM) and super-resolution (SR) have been studied for decades. However, most literature studies these two tasks independently, ignoring the potential benefits of a joint solution. In this paper, aiming at efficient and effective joint demosaicking and super-resolution (JDSR), a well-designed two-stage convolutional neural network (CNN) architecture is proposed. For the first stage, by making use of the sampling-pattern information, a pattern-aware feature extraction (PFE) module extracts features directly from the Bayer-sampled low-resolution (LR) image, while keeping the resolution of the extracted features the same as the input. For the second stage, a dual-branch feature refinement (DFR) module effectively decomposes the features into two components with different spatial frequencies, on which different learning strategies are applied. On each branch of the DFR module, the feature refinement unit, namely, densely-connected dual-path enhancement blocks (DDEB), establishes a sophisticated nonlinear mapping from the LR space to the high-resolution (HR) space. To achieve strong representational power, two paths of transformations and the channel attention mechanism are adopted in DDEB. Extensive experiments demonstrate that the proposed method is superior to the sequential combination of state-of-the-art (SOTA) CDM and SR methods. Moreover, with much smaller model size, our approach also surpasses other SOTA JDSR methods.**

***Index Terms*— Color demosaicking, super-resolution, image restoration, convolutional neural network.**

I. Introduction

DIGITAL color images consist of three color channels. However, for most consumer-grade digital cameras, it is costly to simultaneously measure all three channels for each pixel, and thus a single sensor array is typically utilized in conjunction with a color filter to capture only one color channel per pixel. The most widely used color filter array (CFA) is the Bayer pattern [1]. To restore the other two missing colors for each pixel, color demosaicking (CDM) is required in the image signal processing (ISP) pipeline.

On the other hand, in many applications, a low-cost sensor array often gives rise to low-resolution (LR) images. As a result, super-resolution (SR) is often needed as a postprocessing step to reconstruct high-resolution (HR) output from the given LR image.

Although a large number of CDM and SR methods have been proposed in the last few decades, the two techniques are usually studied independently and applied sequentially in real applications. While treating the two problems separately is much simpler than considering them jointly, the former scheme suffers from three main drawbacks:

1) Suboptimal results. Since both problems are rooted in the limitations of sensors in measuring the spatial/spectral information, they are highly correlated and can both be regarded as inverse problems. Therefore, from a mathematical point of view, it is suboptimal to separate the joint problem into two subproblems and solve them sequentially.
2) Error accumulation. It is very likely that notorious artifacts, such as the zipper effect and false color, will be introduced by the CDM methods to the demosaicked output. As a consequence, these artifacts might mislead the following SR methods, thus resulting in the accumulation and spread of the artifacts in images.
3) A waste of computational and memory resources. For reconstruction-based methods [2]–[11], the "CDM-followed-by-SR" pipeline implies independently exploring prior knowledge for the two tasks on the same image content, which can induce unnecessary repeated computation. For convolutional neural network (CNN)-based methods [12]–[52], the convolutional layers for extracting and refining features cannot be efficiently utilized by both tasks in the sequential pipeline.

Although the concept of joint demosaicking and super-resolution (JDSR) is appealing, it has been under-researched to date. Only a few JDSR schemes have been proposed recently [53]–[55]. In these methods, to directly build a sophisticated mapping from the Bayer LR space to the full-color HR space, a number of basic building blocks, such as the residual blocks [56], the residual in residual dense blocks (RRDB) [26], and the residual channel attention blocks (RCAB) [32], are cascaded. However, the above methods consume quite a large

Manuscript received September 9, 2021; revised October 17, 2021; accepted November 12, 2021. Date of publication November 18, 2021; date of current version July 5, 2022. This work was supported by the National Natural Science Foundation of China (NSFC) under Grant 61761005, Grant 62171145, Grant 61872270, and Grant 61761007. This article was recommended by Associate Editor H. Sun. *(Corresponding author: Kan Chang.)*

Kan Chang, Hengxin Li, and Yufei Tan are with the School of Computer and Electronic Information, Guangxi University, Nanning 530004, China, and also with the Guangxi Key Laboratory of Multimedia Communications and Network Technology, Guangxi University, Nanning 530004, China (e-mail: changkan0@gmail.com; lihengxin_gxu@163.com; jeffrey.yf.tan@gmail.com).

Pak Lun Kevin Ding and Baoxin Li are with the Department of Computer Science, Arizona State University, Tempe, AZ 85287 USA (e-mail: kevinding@asu.edu; baoxin.li@asu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSVT.2021.3129201.

Digital Object Identifier 10.1109/TCSVT.2021.3129201

number of parameters, which is impractical in real-world applications. For example, the model of TENet [54] has more than 20 million (M) parameters, while the smallest version of RDSEN [55] still has more than 6 M parameters. To address this problem, the trade-off between performance and model size should be considered.

Therefore, aiming at efficient and effective JDSR, a well-designed two-stage network structure is proposed in this paper. The architecture for the first stage takes the Bayer-sampled LR image as input, and produces an initial demosaicked result. To efficiently establish an accurate initial result, a pattern-aware feature extraction (PFE) module is presented to directly extract features from the Bayer-sampled LR image, followed by a self-guidance-based refinement (SGR) module to further refine the features. The second-stage structure is responsible for enhancing the initial CDM result and creating a sophisticated nonlinear mapping from the LR space to the HR space. Compared to the first stage, the second stage is much more complex and thus should be allocated with many more parameters. To facilitate the learning of different components in images, we propose a dual-branch feature refinement (DFR) module that sequentially performs decomposition, refinement and fusion of the features. The task of feature refinement in each DFR module is accomplished by the proposed densely-connected dual-path enhancement blocks (DDEB). As shown in Fig. 1, the lightweight version of the proposed two-stage convolutional neural network (TSCNN), namely, TSCNN-L, can achieve a slightly better performance than the latest JDSR scheme RDSEN [55], while the parameters of TSCNN-L are only approximately $1/4$ of that required by RDSEN. Further increasing the depth of the proposed model leads to the heavyweight version of TSCNN called TSCNN-H, which significantly outperforms RDSEN. Overall, the main contributions of this paper are fourfold:

1) For the JDSR task, the Bayer-sampled image is in the LR space, rather than in the HR space. However, the traditional rearrangement-based method further reduces the resolution of the LR input, which could be harmful for restoring details in images. In contrast to the rearrangement-based feature extraction, we attempt to keep the resolution of features unchanged during shallow feature extraction. Therefore, the PFE module is proposed, where the sampling-pattern information is utilized to guide the shallow feature extraction from the Bayer-sampled LR input.
2) An efficient DFR structure is proposed in the second stage. It effectively decomposes the input features into a high-spatial-frequency (HF) component and a low-spatial-frequency (LF) component. Unlike other related works where decomposition is accomplished by using handcrafted design, the decomposition in the DFR module is learned in an end-to-end manner. As it is harder to refine the HF component than the LF component, more resources are allocated to the HF branch. Moreover, the intermediate enhanced LF component is also utilized to support the refinement of the HR component.
3) The dual-path enhancement block (DEB) is presented as the basic building block in the second stage. It contains two paths of transformations with the same topology to capture the underlying frequency information within features. In addition, the channel attention (CA) mechanism is also introduced into DEB to increase the discriminative learning ability of the block. By employing dense connections among the stacked DEBs, we obtain the DDEB unit. Inspired by the hierarchical decomposition in discrete wavelet transform (DWT), the DDEB units are embedded in both DFR branches so that the multi-frequency information in images can be explored hierarchically.
4) Extensive experiments demonstrate that with the smallest model size, our TSCNN surpasses other state-of-the-art (SOTA) methods in terms of both objective and subjective qualities. To facilitate further evaluation, our source code, together with the pretrained models TSCNN-L and TSCNN-H, will be available at https://github.com/Hengxin-Li/TSCNN.

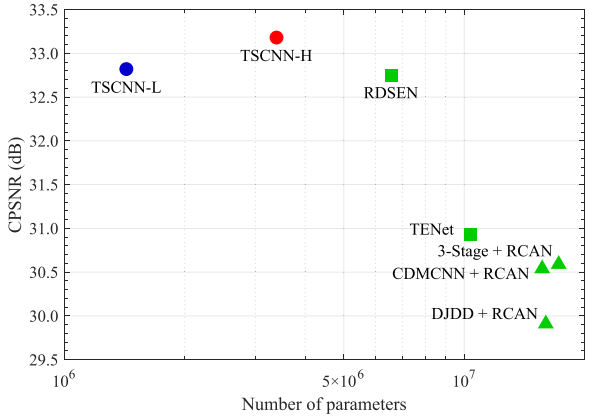


Fig. 1. The trade-off between the average color peak-signal-to-noise ratio (CPSNR) and model size ($\times 2$). The average CPSNR is measured on five widely used datasets: *McM*, *Kodak*, *BSD100*, *Urban100* and *Manga109*.

The rest of this paper is organized as follows: In Section II, the existing approaches for the SR, CDM and JDSR tasks are briefly reviewed. The proposed TSCNN is detailed in Section III. In Section IV, the effectiveness of the proposed approach is verified by extensive experiments. Finally, conclusions are presented in Section V.

II. Related Background

A. Super-Resolution

There are three basic types of SR methods: interpolation-based methods, reconstruction-based methods [2]–[7] and learning-based methods (especially CNN-based methods). Interpolation-based methods are theoretically simple and extremely fast but prone to unsatisfying results. Reconstruction-based approaches consider the SR problem to be an inverse problem and solve it online. Therefore, this type of method suffers from a heavy computational burden. With the help of modern parallel processing techniques, CNN-based methods have been developed very rapidly in recent years. With a much shorter inference time, CNN-based methods

can achieve better performance than reconstruction-based methods.

For CNN-based SR methods, the most important task is to generate a mapping relationship from the LR space to the HR space. Compared with the shallow model SRCNN [12], deeper CNN structures achieve much better performance. However, specific network architectures and training strategies are needed to reduce the training difficulty and further boost performance. For example, to ease the information flow in deep networks, many CNN-based SR methods have applied residual learning [15]–[21]. To utilize the multi-level information within networks, dense connections [22]–[26] have been introduced into many deep structures. To progressively aggregate features efficiently, Ma *et al.* [27] proposed a dense discriminative network, where the aggregation nodes were merged in a tree structure. For video SR, similar to dense connections, Yi *et al.* [28] developed a progressive fusion structure to explore the spatio-temporal information among consecutive frames. To capture different scales of information well, the technique of multi-path learning [57], [58] has also been studied in the SR task [29]–[31]. To explore the correlations among feature channels, the CA mechanism [59] has been widely introduced into SR networks [32], [33]. For more discriminative representations, different types of attention mechanisms have been developed, such as second-order CA [34], joint channel-spatial attention [35], [36] and cross-scale nonlocal attention [37]. In [38], 3D convolutions [60] were used to generate channel-spatial attention. In [39], the attention mechanism was further used to produce combination coefficients in the lattice block to determine the potential linear residual block combinations. In [40], in addition to the mapping from the LR space to the HR space, a dual regression mapping from the HR space back to the LR space was learned. In [41], the intermediate SR results for three image components were separately produced and then merged into the final results.

Besides providing a sophisticated mapping, it is also necessary for CNN-based SR methods to upscale the features to the HR space. The early approaches [12], [15], [22] employed bicubic interpolation to upscale the LR images and then extracted features from the interpolated intermediate images. It is believed that handcrafted-design interpolation can blur the original input images and mislead the network. Hence, in many methods [13], [16], [25], [42], the features were directly extracted from the LR input and then upscaled to HR by using a transposed convolutional layer [61]. In [14], a sub-pixel layer was proposed to learn a set of filters for upscaling the LR feature maps to the HR output. Later, this efficient layer was adopted in many famous CNN structures, such as enhanced deep super-resolution (EDSR) [18], residual dense network (RDN) [24] and residual channel attention network (RCAN) [32].

B. Color Demosaicking

Since there are missing pixels between the sampling positions in each color channel, interpolation-based CDM methods are quite popular. In early approaches, during interpolation, it is frequently assumed that the differences between color channels or the ratios of color channels are constant [62], [63]. However, when these assumptions do not hold, notorious artifacts, such as the zipper effect and false color, may occur. To generate a satisfactory result, a well-known rule is "interpolation along, rather than across edges". Many methods which follow this rule have been presented [64]–[66]. Different from the standard color-difference interpolation, some methods interpolate the residuals between the original samples and the tentatively estimated values, such as minimized-Laplacian residual interpolation (MLRI) [67], iterative residual interpolation (IRI) [68] and adaptive residual interpolation (ARI) [69].

To further improve the quality of the interpolated results, various reconstruction-based methods have been presented. For example, both the inter-channel and intra-channel smoothness were exploited as prior knowledge by [8]. To explore nonlocal self-similarity in images, the models of nonlocal means filtering (NLM), nonlocal adaptive thresholding (NAT) and nonlocal low-rank (NLR) were utilized by [9], [10] and [11], respectively. However, reconstruction-based methods, especially those exploring nonlocal self-similarity in images, suffer from high computational complexity, which hinders the deployment of these methods in real applications.

Many learning-based approaches have also been presented. For example, Khashabi *et al.* proposed a learning-based CDM method based on regression tree fields [70]. By using collaborative representation, the initial interpolated results were progressively refined in [71]. Recently, many CNN-based CDM methods have been developed [43]–[52]. For example, in [44] and [45], a two-stage network and a three-stage network were designed, where the intermediate signals were first established and then further enhanced. In [46], the initial demosaicked result was added to the end of the network so that the network can focus on learning the residual signals. However, these works extract features from the bilinear interpolated results, and thus suffer from the same drawbacks as the SR methods [12], [15], and [22] discussed in Section II-A.

To address this issue, one typical solution is to rearrange a mosaicked image into four matrices [43], [47], [50], [51], which is expressed by

$$\mathbf{F}_p(i, j) = \mathbf{X}(2i + (p \ mod \ 2), 2j + \lfloor \frac{p}{2} \rfloor) \tag{1}$$

where $\mathbf{X}$ is the Bayer-sampled image with a size of $H \times W$, and $\mathbf{F}_p$ denotes the rearranged matrix with a size of $\frac{H}{2} \times \frac{W}{2}$. Note that $i \in [0, \frac{H}{2} - 1]$, $j \in [0, \frac{W}{2} - 1]$; $p = 0, 1, 2, 3$ indicate the four rearranged matrices, respectively. As seen in Equ. (1), the rearrangement leads to a reduction of resolution. As a result, after feature extraction and refinement, it is necessary to upscale the features by using a transposed convolutional layer [61] or a sub-pixel layer [14].

C. Joint Demosaicking and Super-Resolution

Compared to the great number of SR/CDM methods, the bibliography for the JDSR approaches is relatively short. The earliest CNN-based method was proposed in [53], where multiple residual blocks [56] were directly cascaded to establish a nonlinear mapping from the LR manifold to the HR manifold.

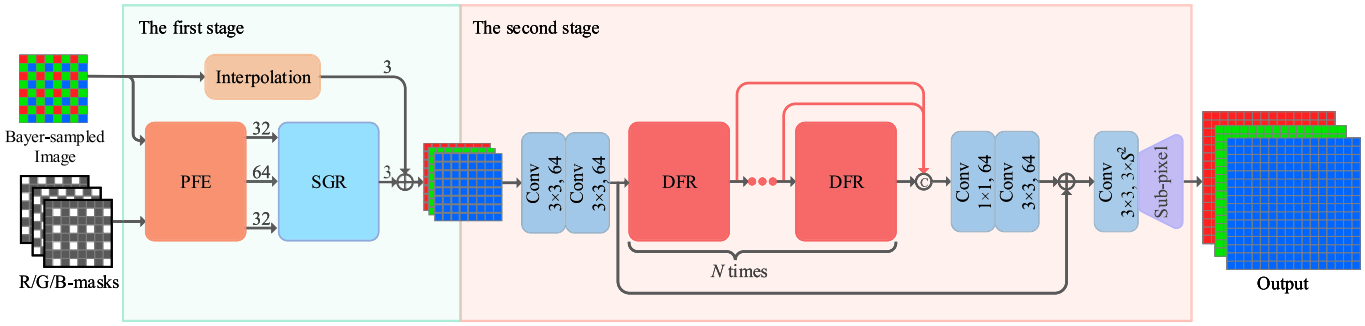


Fig. 2. The overall structure of the proposed TSCNN. The numbers on lines indicate the numbers of feature maps.

Since full-color images are not available, the ground-truth (GT) images in many training datasets, such as the widely used *McM* [10] and *Kodak* [63], are actually demosaicked from the RAW images. To avoid potential artifacts, by using pixel shift technology, Qian *et al.* [54] contributed a realistic dataset called *PixelShift200*. Furthermore, an architecture called TENet, which contains cascaded RRDBs [26], was presented to solve the joint problems [54]. Zhang *et al.* [72] presented a realistic dataset called *SR-RAW*, where the HR GT images were obtained via optical zoom. However, due to the misalignment between the contents captured by different focal lengths, it is necessary to additionally incorporate a contextual bilateral loss [72] during training on *SR-RAW*. In addition to the tasks of CDM and SR, Xu *et al.* [73] proposed a framework to learn the whole ISP pipeline, where a specific network branch was developed to estimate the transformation matrix for color correction. Recently, Xu *et al.* [55] proposed a JDSR network named RDSEN, the building block of which was constructed by introducing dense connections to the RCAB [32]. In front of RDSEN, a pre-demosaicking network (PDNet) was also designed to obtain an intermediate demosaicked result. By jointly training PDNet and RDSEN in an end-to-end manner, the model in [55] achieves the SOTA performance.

However, the performance of the existing JDSR methods relies on the large number of cascaded building blocks, which, unsurprisingly, leads to a relatively large model size. Therefore, the TSCNN model is proposed in this paper. The main differences between TSCNN and the existing two-stage network RDSEN are as follows. 1) The PDNet in RDSEN extracts features at a reduced resolution, and then simply uses 1×1 convolutions to fuse the extracted R, G and B features. For the first-stage structure in TSCNN, the feature extracted by the PFE module has the same resolution as the Bayer-sampled LR image, and the design of the SGR module takes advantage of the fact that the Bayer-sampled G channel has more information than the R and B channels. 2) For the second stage, the well-known RCAB is used as the basic building block of RDSEN, while the building block DEB proposed in this paper is applied in TSCNN. Moreover, in TSCNN, the DDEB unit is embedded in the proposed DFR module, which decomposes the input features into the HF and LF components. Therefore, multi-frequency information in images can be well explored, leading to a highly efficient and effective CNN model.

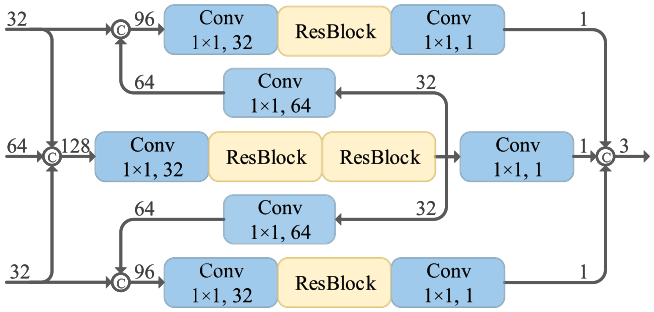


Fig. 3. The structure of the SGR module. The numbers on lines indicate the numbers of feature maps.

III. The Proposed Method

A. The Overall Network Structure

The overall structure of the proposed TSCNN is shown in Fig. 2. Given a $H \times W$ Bayer-sampled LR image, the initial CDM is carried out in the first stage. Then the second stage continues to extract and refine features from the initial CDM result, and finally produces the JDSR output with a dimension of $SH \times SW \times 3$, where S denotes the scale factor.

Specifically, in the first stage, together with the Bayer-sampled LR image, three masks, which indicate the sampling locations in the R, G and B channels, are fed into the TSCNN. To effectively extract features, a PFE module is proposed, the details of which are depicted in Section III-B. The PFE module independently extracts features from the R, G and B signals, and the dimensions of the output R, G and B features are 32, 64 and 32, respectively.

Inspired by [44], [45], after the PFE module, a self-guidance-based refinement (SGR) module is designed to obtain an accurate initial CDM result. As seen in Fig.3, to establish the full G signal, the three inputs are fused by a 1×1 convolutional layer, and then refined by two residual blocks .[1] As the G channel contains more samples, the enhanced G features are used to guide the reconstruction of the R and B channels. With the help of high-quality guidance, it becomes easier to restore the R and B channels. Hence, only one residual block is employed to refine the R/B features. Note that DEB, which is presented later, is not applied in SGR, since

[1]Here we utilize the simplified version of the residual block presented by [18], where batch normalization is removed.

we empirically find that DEB is more effective in refining the deep features, rather than the shallow ones.

To facilitate residual learning [15], the Bayer-sampled LR image is demosaicked in parallel by using an interpolation-based CDM method. To yield a good quality of estimation, the fast and accurate method MLRI [67] is applied. Finally, the MLRI result is added to the SGR output, leading to the output of the first-stage architecture.

In the second stage, features are extracted from the initial CDM result, and then progressively refined. First, two 3×3 convolutional layers are applied for shallow feature extraction. Second, N DFR modules are cascaded to form a precise nonlinear mapping. The DFR module can decompose the input features into two components, each of which is refined by a particular learning strategy. The details of the DFR module are illustrated in Section III-C, and the feature refinement unit in DFR, namely DDEB, is described in Section III-D. To exploit multi-level information in the network, the outputs of all the DFR modules are fused by a 1×1 convolutional layer, and then further refined by a 3×3 convolutional layer. To ease the information flow in the network, a long skip connection, which connects the extracted shallow features to the refined deep ones, is also introduced. Finally, an upsampling module is deployed to reconstruct the JDSR output. It consists of a 3×3 convolutional layer that adjusts the refined features to a dimension of $H \times W \times 3S^2$, and a sub-pixel layer [14] that upsamples the prepared features to a dimension of $SH \times SW \times 3$.

There are two reasons that we use CNN to implement the CDM stage, although this stage poses relatively low computational complexity in the traditional interpolation-based methods. First, CNN-based CDM methods can achieve significantly better performance than interpolation-based methods, which has been proven in other works [44]–[52]. Therefore, using CNN in the first stage can largely avoid the problem of error accumulation mentioned in Section I. Second, as both the first and second stages are implemented by CNN structures, it is possible to find the optimal JDSR results by performing end-to-end training for the whole model.

Although the first stage in TSCNN produces an initial CDM result, TSCNN differs greatly from the approaches that directly combine a CDM subnetwork and an SR subnetwork. First, in TSCNN, the first stage only generates a rough estimation for the full-color LR image, and the feature refinements for the two tasks are jointly carried out in the second stage. By doing so, our model becomes more effective and efficient, as the redundant feature extraction and refinement induced by the "CDM-followed-by-SR" pipeline can be largely reduced. Second, during training, the full-color LR GT image is not provided for intermediate supervision. As a result, the first stage is not forced to produce an intermediate LR result that is degraded from the HR image. Such a setting is robust since the degradation in real applications is usually uncertain due to the interaction between the CDM and SR tasks [54].

Note that for the JDSR task, our two-stage design is superior to the one-stage design [74] where the features are directly extracted from the rearranged Bayer-sampled LR image. With an $H \times W$ Bayer-sampled LR image, the rearranged matrices have a resolution of $\frac{H}{2} \times \frac{W}{2}$, which is large enough for the low-light enhancement task [74]. However, for the JDSR task, a resolution of $\frac{H}{2} \times \frac{W}{2}$ may be too small when compared with the $SH \times SW$ output. Therefore, it is difficult for the one-stage model to capture and depict fine details in images with such a low resolution. However, with a dedicated demosaicking structure, the second-stage structure of TSCNN can refine features with a resolution of $H \times W$, which also benefits the decomposition of the global structures and local details by using the DFR module. Our design is also consistent with the observation in [75], which finds that increasing the resolution of features potentially enables the network to capture more fine-grained patterns.

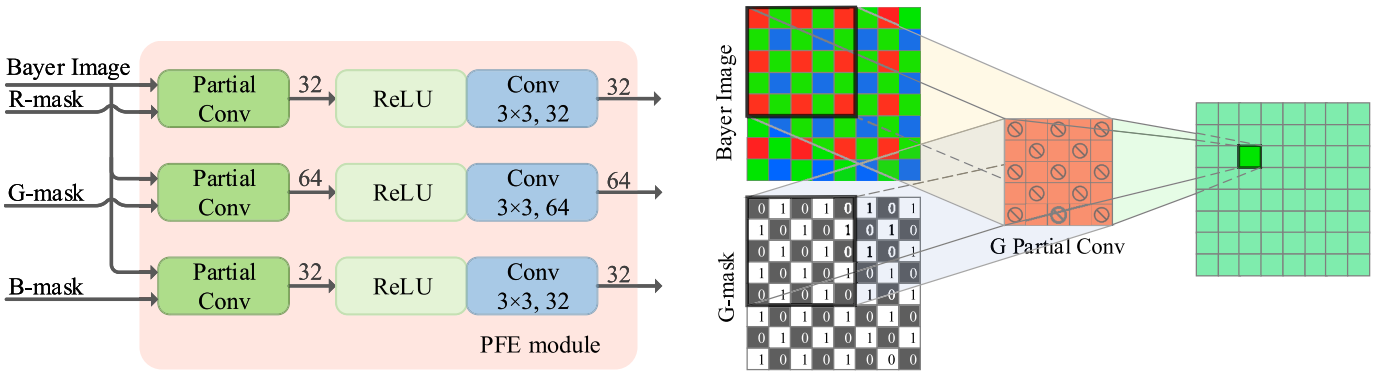
B. The Pattern-Aware Feature Extraction (PFE) Module

As Bayer-sampled images only consist of partial information of R, G and B channels, it is not suitable to directly apply the normal convolutions on the mosaicked images. To address this problem, one solution is to rearrange the $H \times W$ Bayer-sampled image into four $\frac{H}{2} \times \frac{W}{2}$ RGGB matrices, which makes the spatial pattern translation invariant with a period of one pixel [43], [47], [50], [52].

There are two drawbacks to the above method: 1) The rearrangement reduces the resolution of the RGGB matrices to a quarter of the resolution of the Bayer-sampled LR image, which could be harmful for restoring details in images. 2) The rearrangement destroys the original sampling information, which can be useful for designing an effective feature extraction strategy. Therefore, to address the above problems, we propose a PFE module.

The structure of the PFE module is given in Fig. 4 (a), where the R, G and B masks are three $H \times W$ matrices, with "1" and "0" denoting whether there is a valid input at each entry. With the help of the three masks, the PFE module can recognize the sampling patterns in mosaicked images. Consequently, the spatial information can be well preserved and utilized. In the PFE module, there are three shallow feature extraction paths for the three color channels, respectively. On each path, the Bayer-sampled image and the corresponding mask are first fed to a partial convolutional layer [76], and then the output features are activated by a ReLU layer and refined by a 3×3 convolutional layer.

As an example, Fig. 4 (b) shows how to apply the partial convolution to directly extract the G features from a Bayer-sampled image. Before executing the convolution, the element-wise product is calculated between the 5×5 region in the Bayer-sampled image and the co-located region in the G mask. By doing so, those unsampled positions in the G channel are masked out, and the convolution kernels are only "activated" on the unmasked positions. Because the number of unmasked positions varies during the sliding of the convolution window, it is necessary to apply an appropriate scaling on the output, which is similar to the image inpainting task discussed in [76]. However, for the CDM task, the R, G and B masks are regular, and the distance between any two adjacent valid positions is not larger than a 2-pixel distance. Therefore, after one partial convolutional layer, all the "holes" (unsampled pixels) in the R, G or B channel can be filled. As a result, there



(a) The structure of the PFE module (b) Extracting the G features by using partial convolution

Fig. 4. The proposed PFE module. The numbers on lines indicate the numbers of feature maps.

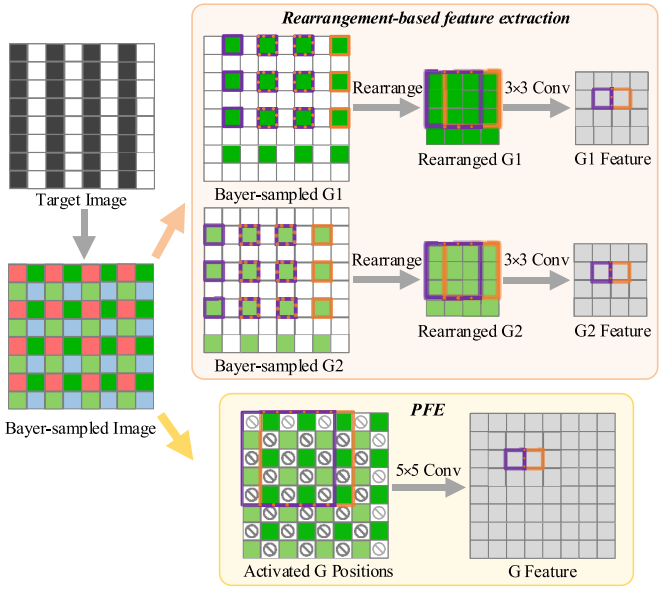


Fig. 5. Comparison between the rearrangement-based feature extraction and the proposed PFE.

is no need to cascade multiple partial convolutional layers and update the masks, which is different from [76].

To better explain the advantages of PFE over traditional rearrangement-based feature extraction, a toy example is given in Fig. 5. For the rearrangement-based feature extraction, performing normal 3×3 convolutions with a stride of 1 on the rearranged G1 and G2 channels is equivalent to performing 3×3 dilated convolutions with a dilation rate of 2 and a stride of 2 on the Bayer-sampled channels. This method is not suitable for precisely recording fine structures in images because 1) local information is missing due to the large gap between the pixels that participate in computation and 2) the coherence of local information is further impaired because of the stride of 2. Assume that black and white stripes appear with a repetition period of two pixels in the target image. As a result, when a convolution kernel moves across the rearranged G1 or G2 channel, the pixels in the receptive field are the same. In contrast, by using PFE, phase changes in the target image can be well detected as the convolution kernel shifts by a stride. In addition, by applying the G-mask to reveal sampling locations, more closely related pixels can take part in computation. Therefore, the incoherence of local information can be largely reduced.

C. The Dual-Branch Feature Refinement (DFR) Module

It is well known that a natural image can be separated into the LF component, which represents the global structures, and the HF component, which depicts the local details. Therefore, it is desirable to process these two types of components individually, so that the network can concentrate on refining different types of features. However, many existing methods decompose the two components by using handcrafted designs. For example, decomposition was explicitly carried out by applying the weighted least squares method, the guided filter, and convolutional sparse coding in [77], [78], and [79], respectively, whereas decomposition was indirectly performed in [80] by providing the two subnetwork branches with LF and HF GT images.

In this paper, a DFR module, which considers decomposing the features into LF and HF components, is proposed. The structure of the DFR module is shown in Fig. 6, where the upper branch and the lower branch are responsible for refining the features for global structures and local details, respectively. It should be noted that instead of using handcrafted designs such as [77]–[80], the decomposition in the proposed DFR module is learned in an end-to-end manner.

Let $\mathbf{X}_I$ be the input feature with a dimension of $H \times W \times 64$. The output of the upper branch $\mathbf{X}_{OU}$ is obtained by

$$\mathbf{X}_{OU} = F_U(F_E(F_E(F_D(\mathbf{X}_I)))) \tag{2}$$

where $F_D(.)$ indicates a downsampling function implemented by applying a 3×3 convolution with a sliding step of 2; $F_E(.)$ denotes the feature enhancement unit DDEB, the structure of which is shown in Fig. 7 (a); $F_U(.)$ stands for the $\times 2$ feature upsampling function by cascading a 1×1 convolutional layer for feature expansion and a sub-pixel layer [14] for aggregating the expanded features.

By performing $F_D(.)$ on $\mathbf{X}_I$, the resolution of the features on the upper branch reduces to $\frac{H}{2} \times \frac{W}{2}$. Hence, if we further apply convolutions on $F_D(\mathbf{X}_I)$, the receptive field in the original feature space is indirectly but effectively enlarged, which is suitable for representing the global structures. Moreover,

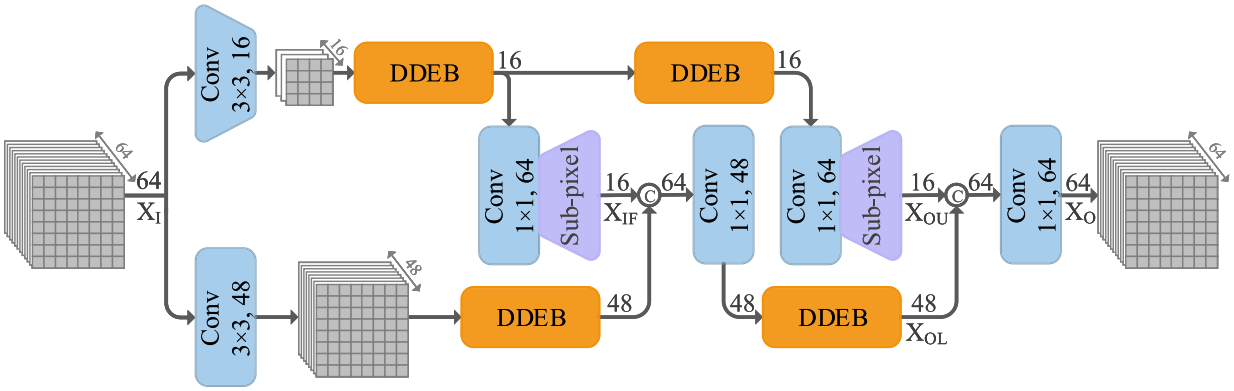


Fig. 6. The structure of the proposed DFR module. The number on each line indicates the number of feature maps.

compared with local details, smooth structures are easier to capture and describe. As a result, it is reasonable to allocate less memory and computational resources to the upper branch. By setting the feature dimension on the upper branch as $\frac{H}{2} \times \frac{W}{2} \times 16$, the required number of network parameters can be largely reduced.

The output of the lower branch $\mathbf{X}_{OL}$ is achieved by

$$\mathbf{X}_{OL} = F_E(F_F([F_E(F_C(\mathbf{X}_I)), \mathbf{X}_{IF}])) \tag{3}$$

where [.] denotes the operation of concatenation; $F_C(.)$ and $F_F(.)$ represent the 3×3 convolutional layer for compressing the input features and the 1×1 convolutional layer for fusing the features, respectively; $\mathbf{X}_{IF}$ denotes the inter-frequency feature with a dimension of $H \times W \times 16$, and it is upsampled from the upper branch by

$$\mathbf{X}_{IF} = F_U(F_E(F_D(\mathbf{X}_I))) \tag{4}$$

On the lower branch, the dimension of features is $H \times W \times 48$, which is more suitable for depicting the local details than the upper branch (although it consumes more parameters and computational resources). However, since the local details are much harder to restore than the global structures, the intermediate extracted features on the upper branch are upsampled and fused with the features on the lower branch to provide more clues for the refinement of features on the lower branch.

Finally, the output of the DFR module, namely $\mathbf{X}_O$, is obtained by using a 1×1 convolutional layer to fuse the concatenated features $\mathbf{X}_{OU}$ and $\mathbf{X}_{OL}$.

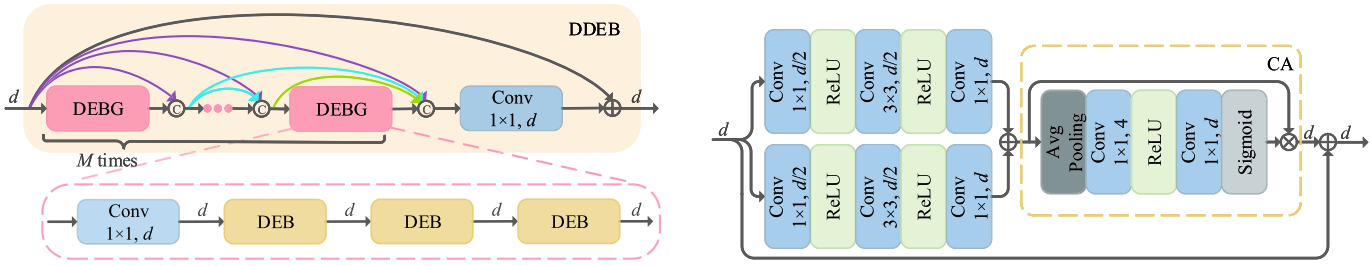
It should be noted that the DFR module differs greatly from the decomposition method proposed in [81], where the decomposition is also learned in an end-to-end manner. For simplicity, the decomposition method in [81] is called "split-and-fusion" in this paper. The main differences between the two approaches are as follows. 1) The definitions of "global features" in the two methods are different. In [81], the global path contains three fully-connected layers. Therefore, the output global features have a resolution of 1×1, which can only reveal very limited global information, such as scene category. However, the resolution of the global features in the DFR module is $\frac{H}{2} \times \frac{W}{2}$, which can depict large structures in images (the visual results are shown in Section IV-B). 2) Due to the usage of the fully-connected layers on the global path of the "split-and-fusion" structure, the resolution of the input feature must be fixed. As a result, the input features should be resized to the predefined resolution before entering the "split-and-fusion" structure. However, the proposed DFR module can directly process images of arbitrary resolution. 3) In the "split-and-fusion" structure, the global features have the same number of channels with the local features. However, in the DFR module, a relatively small number of channels are allocated to the upper branch, making it more suitable for representing large structures rather than fine details. 4) The intermediate global features are used to promote feature extraction on the lower branch of the DFR module, while there is no interaction between the two paths in the "split-and-fusion" structure. The experimental comparison between the two decomposition methods is given in Section IV-B.

D. The Unit of Densely-Connected Dual-Path Enhancement Blocks (DDEB)

To ensure a faithful information recovery, the design of DDEB in the DFR module is of great importance. The structures of DDEB and DEB are given in Fig. 7 (a) and (b), respectively.

In DDEB, M DEB groups (DEBG) are densely connected to exploit the multi-level information and support contiguous memory among different DEBGs. To further facilitate the information flow, the input of DDEB is directly added to the output of the final 1×1 convolutional layer. In each DEBG, a 1×1 convolutional layer is used to fuse the input features, followed by three cascaded DEBs to progressively refine the fused features.

As seen in Fig.7 (b), in addition to the short skip connection, in DEB, two transformation paths with the same topology are designed. At the beginning of each path, the number of feature maps is reduced from d to $d/2$ by using a 1×1 convolutional layer. By doing so, the computational complexity can be largely reduced. Afterward, the compressed features are refined by a 3×3 convolutional layer, and then expanded back to a dimension of d by another 1×1 convolutional layer. After aggregating the two paths, we further integrate a CA unit [59] to rescale the aggregated features. In the JDSR task, along with the high-frequency information, $2/3$ of the channel information is lost during the sampling with a CFA. Therefore,



(a) The structure of the DDEB unit (b) The structure of the DEB

Fig. 7. The proposed DDEB unit and the building block DEB. d is the number of feature maps. For the DDEB units on the upper and the lower branches in DFR, $d = 16$ and $d = 48$, respectively.

in the basic building block, it is desirable to introduce the CA unit to pay extra attention to different channels.

Note that the parameters required by a DEB are far fewer than those required by a normal residual block [56]. More specifically, for a feature dimension of d, a residual block with two 3×3 convolutional layers has $18d^2$ parameters, while the number of parameters required by a DEB is approximately $6.5d^2$ (ignoring the very limited parameters in the CA unit). Therefore, in a DEBG, stacking three DEBs only leads to a number of parameters close to a normal residual block.

It should also be noted that the proposed DEB shares the same strategy of "split-transform-merge" as the block of the aggregated residual transformations (ART) [58], which shows a representation ability more powerful than the residual block [56] and the inception module [57]. Although it has been found that a larger cardinality (which means the number of transformations/paths) can lead to a higher performance [58], the cardinality in DEB is set as 2. The reasons are as follows. 1) The DDEB units are embedded in DFR, where a dual-branch architecture is designed. Therefore, the multi-frequency information in images can be well explored by the "dual-path in dual-branch" network structure, which is quite similar to the spirit of hierarchical decomposition in DWT. As a result, a cardinality larger than 2 can only bring very limited improvement to TSCNN. 2) It is empirically found that a large cardinality can make the required training time of the TSCNN increase dramatically and nonlinearly. Moreover, a complex structure with a large cardinality can also lead to long inference time.

IV. Experimental Results

A. Experimental Settings

We perform experiments on both synthetic and realistic datasets. For the experiments on synthetic datasets, the SR benchmark dataset *DIV2K* [82] is used for training. Note that we do not perform data augmentation on the training set. Two widely used CDM datasets *McM* [10] and *Kodak* [63], together with three well-known SR datasets *BSD100* [83], *Urban100* [84] and *Manga109* [85], are chosen for testing. To generate the mosaicked LR images, we first downsample the original images by using bilinear interpolation and then apply Bayer sampling on the LR images.

For the experiments on the realistic dataset, we use *PixelShift200* [54]. By using pixel shift technology, four color channels are captured at each pixel, including one R, one B and two G channels. As high-quality images are not obtained by demosaicking the RAW images, potential artifacts can be avoided. However, when measuring the objective quality of the results, the two G channels should be averaged. Note that the images in *PixelShift200* are sampled in the linear RGB (linRGB) space. Hence, all the experiments on this dataset are carried out in the linRGB space, and then the outputs are color transformed and gamma corrected into the standard RGB (sRGB) space for display.

Furthermore, we carry out experiments for learning the whole camera ISP pipeline on the dataset presented in [73], where realistic RAW and color training images are obtained by simulating the image degradation process on the well-known dataset *MIT-Adobe FiveK* [86]. For convenience, the dataset presented in [73] is called *JDSR-ISP* in this paper. All the LR images in *JDSR-ISP* are degraded by variable blur kernels and contaminated by heteroscedastic Gaussian noise with random noise parameters. For each training/testing image, a degraded Bayer-sampled LR image in the linRGB space, a degraded full-color LR image in the sRGB space (processed by dcraw[2] and compressed by JPEG), and a full-color HR GT image in the sRGB space are included in the dataset. Given a Bayer-sampled LR image in the linRGB space, the models trained on *JDSR-ISP* are required to directly produce the corresponding full-color HR image in the sRGB space.

Note that in the above datasets, the sampling pattern is predefined as RGGB. For the datasets collected from cameras with other Bayer patterns (GRBG, BGGR, and GBRG) or other CFA patterns such as Fuji X-Trans, the R, G, and B masks for the PFE module should be changed accordingly. However, due to space limitations, those situations are not considered in our experiments. For RAW images in real applications, the sampling pattern information can be directly read from the metadata.

To assess the quality of restored full-color HR images, the color peak-signal-to-noise ratio (CPSNR) [1] and structure similarity (SSIM) index are used. CPSNR is calculated by

$$\mathrm{CPSNR} = 10\log_{10}\frac{255^2}{\frac{1}{3}\sum_{c=R,G,B}\mathrm{MSE}(c)} \tag{5}$$

where $\mathrm{MSE}(c)$ denotes the mean square error (MSE) in the R, G, or B channel. Furthermore, the numbers of parameters

[2]Available at https://www.dechifro.org/dcraw/.

TABLE I
DESCRIPTIONS OF THE VARIANT MODELS IN THE ABLATION STUDY

Name	Description
B-CDM	The baseline network for the CDM task, which is equivalent to the pre-demosaicking network PDNet in [55].
$\mathbb{N}_a$	Introducing the PFE module into B-CDM, while removing the upscaling module in B-CDM.
$\mathbb{N}_b$	Applying the rearrangement-based feature extraction, while using the SGR module to extract features.
$\mathbb{N}_c$	The proposed first-stage (FS) structure.
B-JDSR-L	The lightweight baseline model for the JDSR task, which has the same structure as RDSEN but with fewer RDSEBs [55].
B-JDSR-H	The heavyweight baseline model for the JDSR task, which has more RDSEBs than B-JDSR-L (but still fewer than RDSEN).
$\mathbb{N}_d^L$ / $\mathbb{N}_d^H$	Replacing the PDNet [55] in B-JDSR-L / B-JDSR-H by the proposed FS structure.
$\mathbb{N}_e^L$ / $\mathbb{N}_e^H$	Incorporating the DFR modules into $\mathbb{N}_d^L$ / $\mathbb{N}_d^H$, while utilizing RDSEBs as the building blocks on both branches of DFR.
$\mathbb{N}_f^L$ / $\mathbb{N}_f^H$	Our full TSCNN-L / TSCNN-H architecture.

and floating point operations (FLOPs)[3] required by different models are used to measure the model size and computational complexity, respectively.

Assume that on the upper and lower branches of the DFR module, the numbers of DEBGs in a DDEB unit are M_1 and M_2, respectively, and the number of DFR modules in TSCNN is N. To fully evaluate the performance of TSCNN, two versions of TSCNN are trained, which are denoted as TSCNN-L (lightweight version with $M_1 = 2$, $M_2 = 3$, $N = 3$), and TSCNN-H (heavyweight version with $M_1 = 4$, $M_2 = 6$, $N = 4$). During training, the batch size is 64, and the size of the Bayer-sampled LR image patch is 48×48. ADAM is selected as the optimizer, with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Both versions of TSCNN are trained for 500 epochs, and the initial learning rate is set as 2×10^{-4}, which decreases by half at the 200^{th}, 300^{th} and 400^{th} epochs. All the models are trained from scratch.

When training the TSCNN, for the k-th training patch pair $\{\mathbf{x}^{(k)}, \mathbf{y}^{(k)}\}$, the following loss function is used

$$\mathcal{L}(\Theta) = \frac{1}{N_b} \sum_{k=1}^{N_b} \| \mathcal{F}_{\text{TSCNN}}(\mathbf{x}^{(k)}; \Theta) - \mathbf{y}^{(k)} \|_2^2 \tag{6}$$

where $\mathcal{F}_{\text{TSCNN}}(\cdot)$ represents the function of TSCNN, and Θ denotes the corresponding parameter set; N_b is the batch size.

We implement the two versions of TSCNN with the PyTorch framework. On an Nvidia Tesla V100 GPU, training the TSCNN-L and TSCNN-H models requires approximately 3 days and 6 days, respectively. Our source code and the pretrained models will be available for downloading at https://github.com/Hengxin-Li/TSCNN.

B. Ablation Study and Discussions

To verify the effectiveness of the proposed modules, in this section, a number of variant models are compared on two widely used datasets *McM* [10] and *Kodak* [63]. The 24 images in the *Kodak* dataset have high spectral correlations, while the 18 images in the *McM* dataset are more saturated and contain more sharp structures [10]. As the two datasets contain various scenes and have different characteristics, it is likely that the results and observations obtained in this section can be well generalized into more potential datasets.

[3]Following [29], the number of FLOPs is measured by assuming that the resolution of the full-color HR image is 1280×720. The numbers of parameters and FLOPs reported in this paper are calculated by using the PyTorch tool available at https://github.com/sovrasov/flops-counter.pytorch.

TABLE II
ABLATION STUDY ON THE FIRST-STAGE STRUCTURE FOR CDM TASK

		B-CDM	$\mathbb{N}_a$	$\mathbb{N}_b$	$\mathbb{N}_c$
PFE		×	✓	×	✓
SGR		×	×	✓	✓
Parameters (M)		0.161	0.161	0.151	0.151
FLOPs (G)		18.9	74.2	17.5	69.6
McM	CPSNR	37.40	37.97	37.96	38.21
	SSIM	0.9849	0.9863	0.9867	0.9871
Kodak	CPSNR	40.67	41.29	41.15	41.42
	SSIM	0.9943	0.9948	0.9949	0.9951

1) The Effectiveness of PFE and SGR: Since our first-stage subnetwork is used to generate an initial demosaicked image, to demonstrate the effectiveness of the PFE and SGR modules, we independently evaluate the first-stage structure for the CDM task. In order to make things clearer, we define the compared variant models in Table I, and the average CPSNR/SSIM obtained by these models are shown in Table II. As can be seen, 1) applying the PFE module to B-CDM leads to better performance since the PFE module can directly extract features from the Bayer-sampled image and keep the resolution of features unchanged. 2) $\mathbb{N}_c$ achieves the best performance and requires fewer parameters than B-CDM, proving both the effectiveness and efficiency of the proposed modules. 3) $\mathbb{N}_b$ is inferior to $\mathbb{N}_c$, suggesting that extracting features from the images with reduced resolution can impair the performance.

2) The Effectiveness of DFR and DDEB: For the JDSR task, Tables III and IV present the ablation study on the lightweight and heavyweight variant models, respectively. The compared variant models are described in Table I. For a fair comparison, the number of building blocks in RDSEN [55] is reduced, so that we can obtain two baseline models with the numbers of parameters similar to TSCNN-L and TSCNN-H, respectively. It can be concluded that 1) For the CDM task, our first-stage architecture achieves obvious CPSNR and SSIM improvements over B-CDM. However, for the JDSR task, the improvements of $\mathbb{N}_d^L$ and $\mathbb{N}_d^H$ over B-JDSR-L and B-JDSR-H become less significant. The reason lies in that for the two baseline models, the second-stage architecture, which occupies most of the parameters, can partially compensate for the defects in PDNet [55]. 2) $\mathbb{N}_e^L$ and $\mathbb{N}_e^H$ have significant improvements

TABLE III

ABLATION STUDY FOR JDSR TASK (×2, LIGHTWEIGHT VARIANT MODELS). FS STANDS FOR THE FIRST-STAGE STRUCTURE

		B-JDSR-L	$\mathbb{N}_d^L$	$\mathbb{N}_e^L$	$\mathbb{N}_f^L$
FS		×	✓	✓	✓
DFR		×	×	✓	✓
DDEB		×	×	×	✓
Parameters (M)		1.446	1.432	1.226	1.428
FLOPs (G)		613.1	663.9	544.3	612.7
McM	CPSNR	32.75	32.82	33.10	33.49
	SSIM	0.9471	0.9482	0.9514	0.9548
Kodak	CPSNR	32.88	32.94	33.34	33.68
	SSIM	0.9385	0.9390	0.9422	0.9453

TABLE IV

ABLATION STUDY FOR JDSR TASK (×2, HEAVYWEIGHT VARIANT MODELS). FS STANDS FOR THE FIRST-STAGE STRUCTURE

		B-JDSR-H	$\mathbb{N}_d^H$	$\mathbb{N}_e^H$	$\mathbb{N}_f^H$
FS		×	✓	✓	✓
DFR		×	×	✓	✓
DDEB		×	×	×	✓
Parameters (M)		3.474	3.464	3.235	3.392
FLOPs (G)		1548.5	1599.2	1436.5	1454.4
McM	CPSNR	33.21	33.29	33.50	33.88
	SSIM	0.9520	0.9526	0.9550	0.9573
Kodak	CPSNR	33.48	33.58	33.77	33.98
	SSIM	0.9432	0.9437	0.9461	0.9475

over $\mathbb{N}_d^L$ and $\mathbb{N}_d^H$, respectively, suggesting that using the two branches of architecture to depict the global structure and local details is effective. Moreover, since the number of feature maps on either branch is less than that of the input, the numbers of parameters required by $\mathbb{N}_e^L$ and $\mathbb{N}_e^H$ are smaller than $\mathbb{N}_d^L$ and $\mathbb{N}_d^H$, respectively. 3) The combination of DFR and DDEB units yields the best CPSNR/SSIM in Tables III and IV, which not only demonstrates the effectiveness of the DDEB unit, but also indicates that the multi-frequency information in images/features can be well explored by the proposed hierarchical structure.

3) The Effectiveness of Two-Stage Design: To evaluate the effectiveness of the proposed two-stage design, TSCNN-L and TSCNN-H are compared with the one-stage design [74], where features are extracted and progressively refined with a resolution of $\frac{H}{2} \times \frac{W}{2}$. For a fair comparison, we keep the main bodies of the compared one-stage models the same as the second-stage architectures of TSCNN-L and TSCNN-H, respectively. To facilitate residual learning for the one-stage models, the Bayer-sampled LR image is also demosaicked in parallel by MLRI [67], and then upscaled by bicubic interpolation to make a prediction for the output full-color HR image. Finally, to combine the predicted output and the learned residual, two RCABs [32] and two 3×3 convolutional layers are additionally applied at the end of each one-stage model.

A detailed comparison between the one-stage and two-stage models can be found in Table V, where TS-L and TS-H represent our two-stage models TSCNN-L and TSCNN-H, respectively; OS-L and OS-H stand for the one-stage models whose numbers of parameters are close to TSCNN-L and TSCNN-H, respectively. In addition, some visual results are shown in Fig. 8. As can be seen from Table V and Fig. 8, TS-L and TS-H significantly outperform OS-L and OS-H, respectively. This experiment validates that designing a dedicated demosaicking structure for the JDSR task is necessary.

TABLE V

COMPARISON BETWEEN THE ONE-STAGE AND TWO-STAGE MODELS (×2)

Metric		OS-L / TS-L	OS-H / TS-H
Parameters (M)		1.616 / 1.428	3.472 / 3.392
FLOPs (G)		691.9 / 612.7	1662.3 / 1454.4
McM	CPSNR	31.77 / 33.49	32.05 / 33.88
	SSIM	0.9423 / 0.9548	0.9453 / 0.9573
Kodak	CPSNR	32.01 / 33.68	32.39 / 33.98
	SSIM	0.9323 / 0.9453	0.9359 / 0.9475

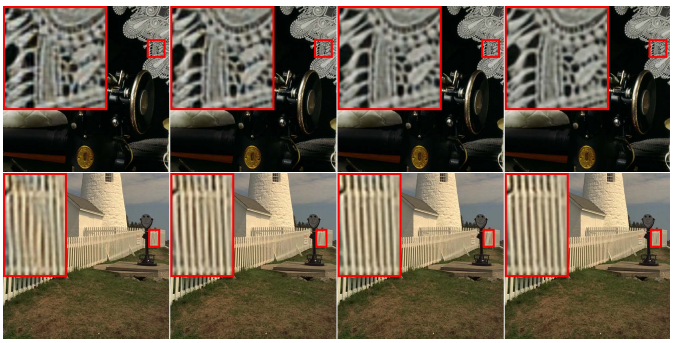
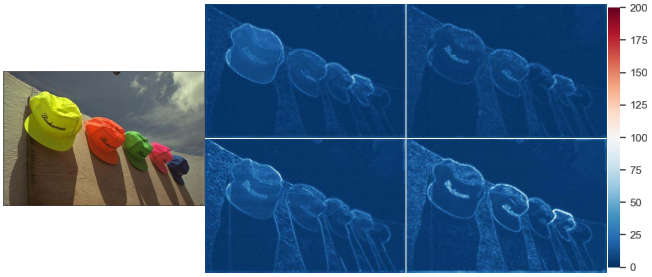


Fig. 8. Visual comparisons among the JDSR results obtained by the one-stage models (OS-L and OS-H) and the two-stage models (TS-L and TS-H) (×2). From left to right: OS-L, OS-H, TS-L, TS-H (zoom in for best view).

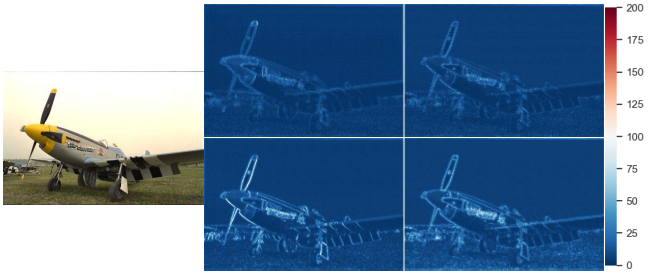
4) The Effectiveness of Our Learning-Based Decomposition: To verify the effectiveness of the proposed learning-based decomposition method, two variant models of TSCNN-L and TSCNN-H are respectively built, where the DFR modules are directly replaced by the "split-and-fusion" structure [81]. For a fair comparison, in these two variant models, DDEB units are applied on the local path of the "split-and-fusion" structure for feature extraction (DDEB is not suitable for the global path of the "split-and-fusion" structure because this path mainly consists of fully-connected layers).

The comparison between the two learning-based decomposition methods can be found in Table VI, where DFR-L and DFR-H are our TSCNN-L and TSCNN-H, respectively; SF-L and SF-H stand for the two variant models with the "split-and-fusion" structure, whose numbers of parameters are close to TSCNN-L and TSCNN-H, respectively. Obviously, with fewer parameters, our DFR module outperforms the "split-and-fusion" structure, which proves the effectiveness of the DFR module.

Moreover, we visualize the output feature maps on the two branches of the DFR module, i.e., $\mathbf{X}_{OU}$ and $\mathbf{X}_{OL}$, in Fig. 9. Note that $\mathbf{X}_{OU}$ and $\mathbf{X}_{OL}$ have 16 and 48 feature maps, respectively. Here we only show the average of the 16 feature maps in $\mathbf{X}_{OU}$ and the average of the 48 feature maps in $\mathbf{X}_{OL}$. As seen in Fig. 9, the upper branch focuses more on describing the global structure, such as shapes and sizes of objects, while the lower branch concentrates more on depicting the local details, such as textures and edges.



(a) *kodim03* from the *Kodak* dataset



(b) *kodim20* from the *Kodak* dataset

Fig. 9. Visualization of the output feature maps on the two branches of the DFR module. In each example, the first and second rows indicate $\mathbf{X}_{OU}$ on the upper branch and $\mathbf{X}_{OL}$ on the lower branch, respectively; the three columns from left to right are the original image, the results of the final DFR modules in TSCNN-L and TSCNN-H, respectively (zoom in for best view).

TABLE VI
COMPARISON BETWEEN TWO DECOMPOSITION METHODS (×2)

Metric		SF-L / DFR-L	SF-H / DFR-H
Parameters (M)		1.965 / 1.428	3.998 / 3.392
FLOPs (G)		589.9 / 612.7	1412.1 / 1454.4
McM	CPSNR	33.28 / 33.49	33.57 / 33.88
	SSIM	0.9535 / 0.9548	0.9551 / 0.9573
Kodak	CPSNR	33.55 / 33.68	33.76 / 33.98
	SSIM	0.9439 / 0.9453	0.9459 / 0.9475

TABLE VII
THE EFFECTS OF THE CARDINALITY IN DEB (×2, LIGHTWEIGHT VARIANT MODELS)

Metric		RCAB	Car. = 1	Car. = 2	Car. = 3
Parameters (M)		1.421	1.432	1.428	1.478
FLOPs (G)		613.8	613.8	612.7	633.0
McM	CPSNR	33.27	33.31	33.49	33.48
	SSIM	0.9533	0.9538	0.9548	0.9544
Kodak	CPSNR	33.60	33.67	33.68	33.70
	SSIM	0.9443	0.9450	0.9453	0.9453

TABLE VIII
COMPARISON BETWEEN APPLYING DEBS AND RESIDUAL BLOCKS IN SGR FOR JDSR TASK (×2)

Metric		DSGR-L / RSGR-L	DSGR-H / RSGR-H
Parameters (M)		1.427 / 1.428	3.391 / 3.392
FLOPs (G)		611.3 / 612.7	1453.0 / 1454.4
McM	CPSNR	33.38 / 33.49	33.65 / 33.88
	SSIM	0.9545 / 0.9548	0.9559 / 0.9573
Kodak	CPSNR	33.68 / 33.68	33.80 / 33.98
	SSIM	0.9454 / 0.9453	0.9465 / 0.9475

This observation clearly demonstrates that the DFR module can effectively decompose the features, which is interesting as we did not use any handcrafted designs or extra supervision during training. Compared with TSCNN-L, the lower branch of the DFR module in TSCNN-H can portray richer details, e.g., the textures on the wall in Fig. 9 (a), the grass under the plane in Fig. 9 (b). Additionally, the deeper structure of TSCNN-H also enables the upper branch of the DFR module to reveal more details than the upper branch in TSCNN-L.

The reasons that such a design can effectively decompose the features lie in its two properties: first, the features on the upper and lower branches have different resolutions and different numbers of channels; second, the intermediate features on the upper branch are used to promote feature extraction on the lower branch. As a result, the lower branch has a more powerful representation ability, making it more suitable for representing local features. Compared with local features, capturing global structures requires a larger receptive field and fewer computational resources. Thus, the structure of the upper branch is the right fit for this task. Therefore, after end-to-end training, the lower branch tends to extract local features, and the upper branch is more likely to convey global information.

5) The Effects of Cardinality in DEB: To verify the effects of the cardinality in DEB, different variant models of TSCNN with different cardinalities are evaluated. However, as mentioned before, a large cardinality makes the training time increase dramatically and nonlinearly, especially for the heavyweight models. For example, when the cardinality is set as 3, training such a variant model of TSCNN-H costs more than 20 days on an Nvidia Tesla V100 GPU. Therefore, we only show the results obtained by different variant models of TSCNN-L in Table VII, where "car." is short for cardinality. Note that when the cardinality is 1, the structure of DEB is very similar to that of RCAB [32]. The difference between "RCAB" and "car. = 1" lies in that in each building block, the former structure has two 3×3 convolutional layers, while the latter contains one 3×3 and two 1×1 convolutional layers. As a larger cardinality leads to a lower number of parameters in DEB, for a fair comparison, we adjust the feature dimension d and the number of DEBs in DEBG, so that all the variant models have similar numbers of parameters.

It can be concluded from Table VII that: 1) The results of "car. = 2" are superior to "car. = 1", especially on the *McM* dataset which contains rich details and edges. 2) The performance of "car. = 3" is close to "car. = 2". Therefore, considering the long training and inference times required by a large cardinality, setting the cardinality in DEB as 2 is reasonable. 3) Despite different cardinalities, all versions of DEB are more effective than RCAB.

6) The Effects of Applying DEBs in SGR: We further conduct experiments on variant models of TSCNN-L and TSCNN-H to assess the effects of applying DEBs in SGR. The results are shown in Table VIII, where DSGR-L and DSGR-H stand for the lightweight and heavyweight variant models which apply DEBs in SGR, respectively; RSGR-L and RSGR-H are the original TSCNN-L and TSCNN-H, respectively. Since a residual block has many more parameters

TABLE IX

QUANTITATIVE COMPARISONS AMONG DIFFERENT MODELS ON FIVE SYNTHETIC DATASETS (AVERAGE CPSNR (DB) / SSIM). THE NUMBERS IN RED AND BLUE INDICATE THE BEST AND THE SECOND-BEST METHODS, RESPECTIVELY

Method	Scale	*McM*	*Kodak*	*BSD100*	*Urban100*	*Manga109*	Average
ARI + RCAN	×2	30.49 / 0.9144	29.34 / 0.8758	28.39 / 0.8458	26.42 / 0.8689	28.67 / 0.9316	28.66 / 0.8873
DJDD + RCAN	×2	30.93 / 0.9263	30.50 / 0.9031	29.46 / 0.8790	28.30 / 0.9045	30.34 / 0.9466	29.91 / 0.9119
CDMCNN + RCAN	×2	31.71 / 0.9343	30.81 / 0.9060	29.63 / 0.8811	28.62 / 0.9084	31.91 / 0.9580	30.54 / 0.9176
3-Stage + RCAN	×2	31.72 / 0.9345	30.84 / 0.9070	29.71 / 0.8825	28.76 / 0.9099	31.91 / 0.9579	30.59 / 0.9184
TENET	×2	31.79 / 0.9374	31.94 / 0.9301	30.89 / 0.9153	28.94 / 0.9231	31.07 / 0.9510	30.93 / 0.9314
RDSEN	×2	33.47 / 0.9540	33.64 / 0.9450	32.15 / 0.9308	31.00 / 0.9448	33.48 / 0.9682	32.75 / 0.9486
TSCNN-L (ours)	×2	33.49 / 0.9548	33.68 / 0.9453	32.20 / 0.9315	31.12 / 0.9460	33.60 / 0.9692	32.82 / 0.9494
TSCNN-H (ours)	×2	**33.88 / 0.9573**	**33.98 / 0.9475**	**32.41 / 0.9339**	**31.55 / 0.9490**	**34.07 / 0.9713**	**33.18 / 0.9518**
ARI + RCAN	×3	27.88 / 0.8514	27.36 / 0.7950	26.21 / 0.7545	24.23 / 0.7919	25.76 / 0.8766	26.29 / 0.8139
DJDD + RCAN	×3	27.98 / 0.8603	27.80 / 0.8126	26.64 / 0.7766	25.05 / 0.8182	26.20 / 0.8840	26.73 / 0.8303
CDMCNN + RCAN	×3	28.56 / 0.8713	28.03 / 0.8165	26.78 / 0.7798	25.26 / 0.8236	27.58 / 0.9071	27.24 / 0.8397
3-Stage + RCAN	×3	28.59 / 0.8719	28.04 / 0.8176	26.84 / 0.7817	25.38 / 0.8265	27.57 / 0.9069	27.28 / 0.8409
TENET	×3	28.63 / 0.8773	28.58 / 0.8469	27.31 / 0.8150	25.41 / 0.8420	27.22 / 0.8999	27.43 / 0.8562
RDSEN	×3	29.84 / 0.8999	29.65 / 0.8712	28.09 / 0.8365	27.07 / 0.8797	29.25 / 0.9304	28.78 / 0.8835
TSCNN-L (ours)	×3	29.86 / 0.9014	29.66 / 0.8710	28.09 / 0.8368	27.05 / 0.8798	29.33 / 0.9326	28.80 / 0.8843
TSCNN-H (ours)	×3	**30.09 / 0.9037**	**29.79 / 0.8734**	**28.21 / 0.8384**	**27.40 / 0.8852**	**29.72 / 0.9360**	**29.04 / 0.8873**
ARI + RCAN	×4	26.03 / 0.7905	25.57 / 0.7312	24.80 / 0.6784	22.42 / 0.7091	23.42 / 0.8116	24.45 / 0.7442
DJDD + RCAN	×4	26.24 / 0.8034	25.90 / 0.7504	25.19 / 0.7028	23.26 / 0.7472	23.80 / 0.8236	24.88 / 0.7655
CDMCNN + RCAN	×4	26.69 / 0.8160	26.15 / 0.7552	25.32 / 0.7069	23.43 / 0.7534	24.92 / 0.8534	25.30 / 0.7770
3-Stage + RCAN	×4	26.72 / 0.8164	26.18 / 0.7560	25.37 / 0.7085	23.53 / 0.7567	24.90 / 0.8533	25.34 / 0.7782
TENET	×4	26.73 / 0.8173	26.59 / 0.7824	25.75 / 0.7407	23.47 / 0.7610	24.67 / 0.8329	25.44 / 0.7869
RDSEN	×4	28.14 / 0.8586	27.78 / 0.8192	26.56 / 0.7728	25.18 / 0.8251	26.97 / 0.8942	26.93 / 0.8340
TSCNN-L (ours)	×4	28.11 / 0.8598	27.78 / 0.8204	26.59 / 0.7751	25.21 / 0.8265	27.05 / 0.8979	26.95 / 0.8359
TSCNN-H (ours)	×4	**28.37 / 0.8641**	**27.94 / 0.8234**	**26.68 / 0.7777**	**25.42 / 0.8319**	**27.49 / 0.9038**	**27.18 / 0.8402**

TABLE X

THE NUMBERS OF PARAMETERS, FLOPS AND INFERENCE TIME OF DIFFERENT NETWORKS (×2). THE NUMBERS IN RED AND BLUE INDICATE THE LIGHTEST/FASTEST AND THE SECOND-LIGHTEST/FASTEST METHODS, RESPECTIVELY

Metric	DJDD + RCAN	CDMCNN + RCAN	3-Stage + RCAN	TENet	RDSEN	TSCNN-L (ours)	TSCNN-H (ours)
Parameters (M)	16.006	15.674	17.238	10.345	6.563	**1.428**	3.392
FLOPs (G)	7143.7	7183.4	7904.8	2984.4	2971.9	**612.7**	1454.4
Inference Time (s)	0.6106	0.6140	0.6601	0.2400	0.2310	**0.2004**	0.5084

than a DEB, the feature dimension and the number of DEBs in DSGR-L/DSGR-H are adjusted to achieve a model size similar to TSCNN-L/TSCNN-H. As shown in Table VIII, DSGR-L is close to RSGR-L on *Kodak* but has a 0.11 dB drop in CPSNR on *McM*. Compared with RSGR-H, obvious performance degradation can be observed on the results achieved by DSGR-H. Moreover, as each DEB contains two paths, embedding DEBs in SGR results in a very complex structure. We find that the training time of DSGR-L/DSGR-H is obviously longer than that of RSGR-L/RSGR-H. Therefore, it is better to deploy residual blocks in SGR for shallow feature extraction.

C. Experiments on Synthetic Datasets

To demonstrate the effectiveness of TSCNN, several SOTA methods are compared. Apart from the recently proposed JDSR networks TENet [54] and RDSEN [55], four brute-force schemes (by directly combining a CDM method and an SR network) are also considered, including ARI [69] + RCAN [32], DJDD [43] + RCAN [32], CDMCNN [44] + RCAN [32] and 3-Stage [45] + RCAN [32]. For a fair comparison, all the compared methods are retrained by using the same settings as described in Section IV-A. Note that the originally released model of TENet has over 20 M parameters. To obtain a magnitude of the model size similar to other compared methods, a 10-M-parameter version of TENet is retrained in our experiment.

The quantitative comparisons on the five synthetic datasets are provided in Table IX. We find that: 1) ARI + RCAN is significantly inferior to the other three brute-force schemes, which demonstrates the advantages of the CNN-based CDM methods over the traditional CDM methods. 2) The joint solutions are significantly superior to the brute-force schemes, which suggests that jointly considering the CDM and SR tasks is necessary. 3) RDSEN obviously outperforms TENet. In TENet, the SR subnetwork shares the same structure as the CDM subnetwork, thus resulting in inefficient feature extraction. In contrast, feature refinement is mainly carried

TABLE XI

QUANTITATIVE COMPARISONS AMONG DIFFERENT MODELS ON THE REALISTIC DATASET *PixelshiftTest* (×4). THE NUMBERS IN RED AND BLUE INDICATE THE BEST AND THE SECOND-BEST METHODS, RESPECTIVELY

Noise level	Metric	ARI + RCAN	DJDD + RCAN	CDMCNN + RCAN	3-Stage + RCAN	TENet	RDSEN	TSCNN-L	TSCNN-H
$\sigma = 5$	CPSNR	38.97	39.73	39.29	39.80	39.90	40.76	41.00	**41.36**
	SSIM	0.9713	0.9756	0.9736	0.9757	0.9749	0.9775	0.9778	**0.9791**
$\sigma = 10$	CPSNR	35.73	37.92	37.09	37.54	39.30	39.59	39.97	**40.28**
	SSIM	0.9599	0.9668	0.9650	0.9670	0.9727	0.9745	0.9743	**0.9758**
$\sigma = 15$	CPSNR	33.37	36.62	36.28	36.52	36.40	37.93	37.99	**38.42**
	SSIM	0.9507	0.9614	0.9612	0.9626	0.9646	0.9691	0.9675	**0.9695**

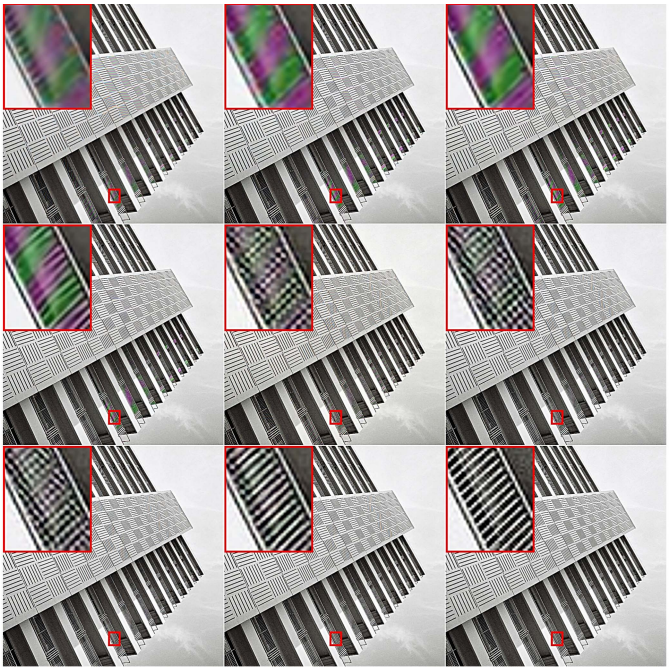


Fig. 10. Results of *img92* from *Urban100* (×2). From left to right and top to bottom: DJDD + RCAN, CDMCNN + RCAN, 3-Stage + RCAN, TENet, RDSEN, TSCNN-L, TSCNN-H, original image (zoom in for best view).

out in the second subnetwork in RDSEN. Therefore, the deep features can be jointly explored for both tasks. 4) The proposed TSCNN-L achieves a performance slightly better than RDSEN, while TSCNN-L only uses 1/4 of the parameters required by RDSEN (the detailed comparison of model size can be found in Table X). By further increasing the depth, the TSCNN-H which has only 3.4 M parameters can consistently provide the best quality of results.

Fig. 10 shows visual comparison among different methods (the results of ARI + RCAN are omitted due to space limitations). We can see that: 1) The brute-force schemes produce unsatisfactory outputs. 2) By jointly considering both tasks, the networks of TENet and RDSEN can provide better results than the four brute-force schemes. However, obvious artifacts can still be found in the results of TENet and RDSEN. 3) Our TSCNN-L can avoid the above artifacts to a great extent and preserve more fine details and structures, which demonstrates the positive effects brought by the proposed modules and units. At the cost of a larger number of parameters, TSCNN-H can deliver even more faithful results than TSCNN-L.

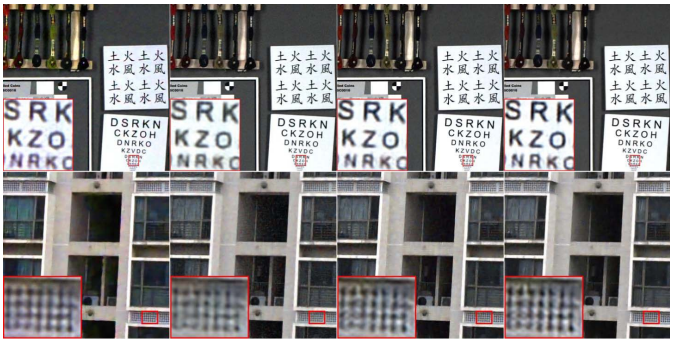


Fig. 11. Results of the images from *Pixelshift200* (×4, $\sigma = 10$). From left to right: TENet, RDSEN, TSCNN-L, TSCNN-H (zoom in for best view).

D. Experiments on the Realistic Dataset

Experiments are further carried out on the realistic dataset *PixelShift200* [54], where the scale factor of ×4 is evaluated, and the Bayer-sampled LR images are contaminated by the Gaussian noise with $\sigma = 5$, 10 and 15. All the compared models are retrained on *PixelShift200*. Since the traditional CDM method ARI [69] cannot remove noise, the denoising method BM3D [87] is utilized after ARI.

The CPSNR and SSIM values obtained by different models on the testing set *PixelshiftTest* are provided in Table XI. TSCNN-L obtains results on par with RDSEN [55], while TSCNN-H gains significant improvements over RDSEN in terms of both CPSNR and SSIM. The above trend is similar to what we can find on the synthetic datasets.

The qualitative comparisons for the two images from *PixelShift200* are provided in Fig.11 (the results of brute-force schemes are omitted due to space limitations). We find that RDSEN produces relatively blurred images, while significant false color occurs in the results of TENet [54]. In addition, noticeable ringing artifacts can be found in the results of TENet and RDSEN. In contrast, both TSCNN-L and TSCNN-H preserve more details and sharper edges, and are less prone to ringing artifacts. The above observation clearly proves the advantages of the proposed two TSCNN models over TENet and RDSEN.

E. Extension to Learn the Camera ISP Pipeline

In the typical camera ISP pipeline, noise reduction is highly related to CDM and SR, as all the three image restoration tasks aim to recover clear details and structures in images. Additionally, the remaining processing steps in the ISP pipeline,

including white balance, color space transform, gamma correction, tone mapping, etc. [88], are applied to properly adjust the colors and lighting. To integrate the TSCNN into the ISP pipeline, one solution is to directly replace CDM, SR and noise reduction by TSCNN, and then independently carry out the remaining processing steps on the TSCNN output, which is similar to the experiments conducted on the *PixelShift200* dataset in the previous part. Another solution is to extend the TSCNN model to directly learn the whole camera ISP pipeline, which is similar to [72], [73], [89], [90].

The experiments for learning the camera ISP pipeline are carried out on the *JDSR-ISP* [73] dataset, where a scale factor of $\times 2$ is evaluated. To learn the camera ISP pipeline, our TSCNN-L is extended to a new model called TSCNN-ISP, whose structure is the same as TSCNN-L except that the result of the interpolation-based CDM method is replaced by the degraded full-color LR image in the sRGB space. TSCNN-ISP is retrained by taking the Bayer-sampled LR images in the linRGB space and the corresponding full-color HR images in the sRGB space as input and output, respectively.

TSCNN-ISP is compared with the dual-CNN model [73], where one branch is used to recover fine details and small structures, while the other branch is used for color corrections. In addition, the SR model RCAN [32] is also compared, which is retrained on the *JDSR-ISP* dataset by taking the degraded full-color LR images in the linRGB space as input.

The quantitative results on the *JDSR-ISP* dataset are presented in Table XII, and the visual results are shown in Fig. 12. It can be concluded that: 1) With the smallest model size, TSCNN-ISP surpasses dual-CNN. This result demonstrates that our TSCNN structure can learn the ISP pipeline well. Unlike the dual-CNN model which has a specific color branch to recover high-fidelity colors, color corrections are implicitly learned in TSCNN-ISP. Therefore, TSCNN still has great potential to achieve even better performance if a specific structure is designed for color corrections. 2) Dual-CNN and TSCNN-ISP are both superior to RCAN, which also indicates that jointly performing CDM and SR has more advantages.

F. Model Size, Computational Burden and Inference Time

To better illustrate the trade-off between image quality and model size, the obtained average CPSNR on the five synthetic datasets and the required parameters by different models at a scale factor of $\times 2$ are plotted in Fig. 1. Furthermore, the detailed numbers of parameters, FLOPs and inference time[4] cost by different approaches are listed in Table X (ARI is omitted as it is not a CNN-based method). Apparently, TSCNN-L and TSCNN-H have the fewest parameters, the lowest computational burden, and the best trade-off between the performance and the model size. Due to the complex structure of TSCNN, the inference time of TSCNN-H is longer than RDSEN and TENet. However, TSCNN-L is still the fastest among all the tested methods. The very appealing efficiency of the TSCNN is mainly attributed to the following reasons:

[4]The average inference time of an image is measured on the *Urban100* dataset. The resolutions of the HR images in this dataset range from 560×1024 to 1024×1024.

TABLE XII
QUANTITATIVE COMPARISON ON THE *JDSR-ISP* DATASET FOR THE TASK OF ISP LEARNING ($\times 2$)

Metric	RCAN [32]	dual-CNN [73]	TSCNN-ISP
Parameters (M)	15.445	3.000	1.428
FLOPs (G)	7077.6	1011.3	612.7
CPSNR (dB)	29.98	30.63	30.78
SSIM	0.8525	0.8672	0.8698

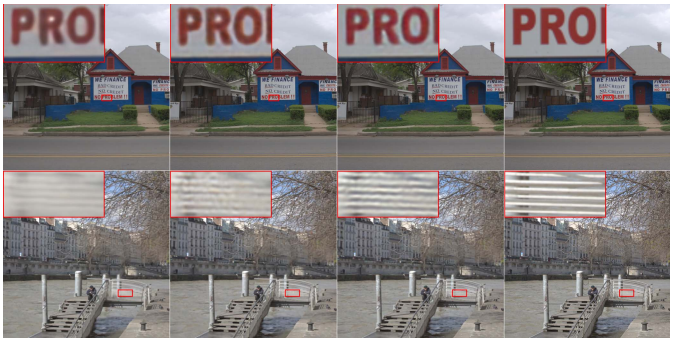


Fig. 12. Results of the images from *JDSR-ISP* [73] ($\times 2$). From left to right: RCAN, dual-CNN, TSCNN-ISP, original image (zoom in for best view).

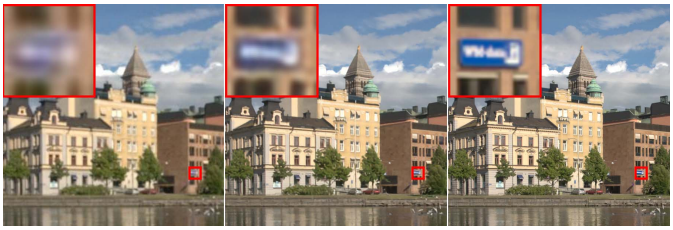


Fig. 13. Failure case. From left to right: degraded full-color LR image, output of TSCNN-ISP, original image. The three images are in the sRGB space. A minor color shift exists in the result of TSCNN-ISP.

1) The dual-branch architecture of the DFR module has a smaller number of parameters than a single branch. Moreover, the low feature resolution on the upper branch of the DFR module leads to a relatively low number of FLOPs. 2) A DEB has far fewer parameters than a normal residual block. As a result, we can construct a deeper network by cascading DEBs, which is beneficial for establishing a more sophisticated nonlinear mapping. In addition, the "dual-path in dual-branch" structure can efficiently explore the multi-frequency information in features, which is also helpful for achieving a satisfactory result with fewer parameters. 3) Since the Bayer-sampled image is in the LR space and the first-stage architecture is rather shallow, compared with the rearrangement-based strategy, the number of FLOPs introduced by the proposed first-stage architecture does not increase much.

G. Limitations

As mentioned in Section IV-E, when learning the whole ISP pipeline, color corrections are implicitly learned in the simple extension of TSCNN. A failure example is shown in Fig. 13, where a minor color shift exists in the result of TSCNN-ISP. In the future, we will focus on developing a specific structure

for effectively correcting colors, so that the TSCNN can be well qualified for learning the ISP pipeline.

V. Conclusion

In this paper, a two-stage CNN model is proposed for the JDSR task. In the first stage, the sampling-pattern information is utilized in the PFE module to assist the shallow feature extraction to obtain an unreduced resolution of extracted features. In the second stage, the DFR module decomposes the features into two components with different spatial frequencies, and then refines the decomposed components by using different learning strategies. In the basic building block, two transformations are carried out to better describe the underlying frequency information, followed by a CA unit rescaling the features according to the importance of different feature maps. By embedding the DDEB units in the DFR module, the "dual-path in dual-branch" network structure can explore the multi-frequency information hierarchically. Through ablation study, we find that all the proposed modules/units contribute to the high-quality results of the TSCNN model. When compared with other SOTA methods, our TSCNN achieves significantly better performance, while consuming much fewer parameters and FLOPs.

Acknowledgment

Some of the experiments were carried out on the High-Performance Computing Platform of Guangxi University. The authors thank Guocheng Qian from SenseTime Research for discussing the usage of the *PixelShift200* dataset with them. They also thank Xuan Xu from West Virginia University for sharing their source code with them.

References

[1] D. Menon and G. Calvagno, "Color image demosaicking: An overview," *Signal Process., Image Commun.*, vol. 26, nos. 8–9, pp. 518–533, Oct. 2011.

[2] W. Dong, L. Zhang, G. Shi, and X. Wu, "Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization," *IEEE Trans. Image Process.*, vol. 20, no. 7, pp. 1838–1857, Jul. 2011.

[3] W. Dong, L. Zhang, G. Shi, and X. Li, "Nonlocally centralized sparse representation for image restoration," *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1620–1630, Apr. 2013.

[4] H. Chen, X. He, L. Qing, and Q. Teng, "Single image super-resolution via adaptive transform-based nonlocal self-similarity modeling and learning-based gradient regularization," *IEEE Trans. Multimedia*, vol. 19, no. 8, pp. 1702–1717, Aug. 2017.

[5] J. Liu, W. Yang, X. Zhang, and Z. Guo, "Retrieval compensated group structured sparsity for image super-resolution," *IEEE Trans. Multimedia*, vol. 19, no. 2, pp. 302–316, Feb. 2017.

[6] K. Chang, P. L. K. Ding, and B. Li, "Single image super resolution using joint regularization," *IEEE Signal Process. Lett.*, vol. 25, no. 4, pp. 596–600, Apr. 2018.

[7] K. Chang, X. Zhang, P. L. K. Ding, and B. Li, "Data-adaptive low-rank modeling and external gradient prior for single image super-resolution," *Signal Process.*, vol. 161, pp. 36–49, Aug. 2019.

[8] D. Menon and G. Calvagno, "Regularization approaches to demosaicking," *IEEE Trans. Image Process.*, vol. 18, no. 10, pp. 2209–2220, Oct. 2009.

[9] A. Buades, B. Coll, J.-M. Morel, and C. Sbert, "Self-similarity driven color demosaicking," *IEEE Trans. Image Process.*, vol. 18, no. 6, pp. 1192–1202, Jun. 2009.

[10] X. Wu, "Color demosaicking by local directional interpolation and nonlocal adaptive thresholding," *J. Electron. Imag.*, vol. 20, no. 2, Apr. 2011, Art. no. 023016.

[11] K. Chang, P. L. K. Ding, and B. Li, "Color image demosaicking using inter-channel correlation and nonlocal self-similarity," *Signal Process., Image Commun.*, vol. 39, pp. 264–279, Nov. 2015.

[12] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2015.

[13] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proc. 14th Eur. Conf. Comput. Vis. (ECCV)*, Amsterdam, The Netherlands, Oct. 2016, pp. 391–407.

[14] W. Shi *et al.*, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1874–1883.

[15] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1646–1654.

[16] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep Laplacian pyramid networks for fast and accurate super-resolution," in *Proc. Int. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 5835–5843.

[17] C. Ledig *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 105–114.

[18] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. Comput. Vis. Pattern Recognit. Workshops*, Jul. 2017, pp. 1132–1140.

[19] N. Ahn, B. Kang, and K.-A. Sohn, "Fast, accurate, and lightweight super-resolution with cascading residual network," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Munich, Germany, Sep. 2018, pp. 256–272.

[20] Y. Wang, L. Wang, H. Wang, and P. Li, "Resolution-aware network for image super-resolution," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 5, pp. 1259–1269, May 2019.

[21] F. Li, H. Bai, and Y. Zhao, "FilterNet: Adaptive information filtering network for accurate and fast image super-resolution," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 6, pp. 1511–1523, Jun. 2020.

[22] Y. Tai, J. Yang, X. Liu, and C. Xu, "MemNet: A persistent memory network for image restoration," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 4549–4557.

[23] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 4809–4817.

[24] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 2472–2481.

[25] M. Haris, G. Shakhnarovich, and N. Ukita, "Deep back-projection networks for super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 1664–1673.

[26] X. Wang *et al.*, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *Proc. Eur. Conf. Comput. Vis. Workshops (ECCVW)*, Sep. 2018, pp. 63–79.

[27] J. Ma, X. Wang, and J. Jiang, "Image superresolution via dense discriminative network," *IEEE Trans. Ind. Electron.*, vol. 67, no. 7, pp. 5687–5695, Jul. 2020.

[28] P. Yi, Z. Wang, K. Jiang, J. Jiang, T. Lu, and J. Ma, "A progressive fusion generative adversarial network for realistic and consistent video super-resolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Dec. 3, 2020, doi: 10.1109/TPAMI.2020.3042298.

[29] J. Li, F. Fang, K. Mei, and G. Zhang, "Multi-scale residual network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Munich, Germany, Sep. 2018, pp. 527–542.

[30] K. Chang, M. Li, P. L. K. Ding, and B. Li, "Accurate single image super-resolution using multi-path wide-activated residual network," *Signal Process.*, vol. 172, Jul. 2020, Art. no. 107567.

[31] J. Li, F. Fang, J. Li, K. Mei, and G. Zhang, "MDCN: Multi-scale dense cross network for image super-resolution," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 7, pp. 2547–2561, Jul. 2021.

[32] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Munich, Germany, Sep. 2018, pp. 294–310.

[33] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, "Residual non-local attention networks for image restoration," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019, pp. 1–18.

[34] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang, "Second-order attention network for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 11065–11074.

[35] Y. Hu, J. Li, Y. Huang, and X. Gao, "Channel-wise and spatial feature modulation network for single image super-resolution," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 11, pp. 3911–3927, Nov. 2020.

[36] J. Liu, W. Zhang, Y. Tang, J. Tang, and G. Wu, "Residual feature aggregation network for image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2356–2365.

[37] Y. Mei, Y. Fan, Y. Zhou, L. Huang, T. S. Huang, and H. Shi, "Image super-resolution with cross-scale non-local attention and exhaustive self-exemplars mining," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 5689–5698.

[38] B. Niu *et al.*, "Single image super-resolution via a holistic attention network," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Glasgow, U.K., Aug. 2020, pp. 191–207.

[39] X. Luo, Y. Xie, Y. Zhang, Y. Qu, C. Li, and Y. Fu, "LatticeNet: Towards lightweight image super-resolution with lattice block," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Glasgow, U.K., Aug. 2020, pp. 272–289.

[40] Y. Guo *et al.*, "Closed-loop matters: Dual regression networks for single image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 5406–5415.

[41] P. Wei *et al.*, "Component divide-and-conquer for real-world image super-resolution," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Glasgow, U.K., Aug. 2020, pp. 101–117.

[42] Z. Hui, X. Wang, and X. Gao, "Fast and accurate single image super-resolution via information distillation network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 723–731.

[43] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand, "Deep joint demosaicking and denoising," *ACM Trans. Graph.*, vol. 35, no. 6, p. 191, 2016.

[44] R. Tan, K. Zhang, W. Zuo, and L. Zhang, "Color image demosaicking via deep residual learning," in *Proc. IEEE Int. Conf. Multimedia Expo. (ICME)*, Jul. 2017, pp. 793–798.

[45] 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. (ICIP)*, Oct. 2018, pp. 2177–2181.

[46] 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.

[47] T. Huang, F. F. Wu, W. Dong, G. Shi, and X. Li, "Lightweight deep residue learning for joint color image demosaicking and denoising," in *Proc. 24th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2018, pp. 127–132.

[48] F. Kokkinos and S. Lefkimmiatis, "Deep image demosaicking using a cascade of convolutional residual denoising networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Munich, Germany, Sep. 2018, pp. 317–333.

[49] F. Kokkinos and S. Lefkimmiatis, "Iterative joint image demosaicking and denoising using a residual denoising network," *IEEE Trans. Image Process.*, vol. 28, no. 8, pp. 4177–4188, Mar. 2019.

[50] Q. Kang, Y. Fu, and H. Huang, "Deep color image demosaicking with feature pyramid channel attention," in *Proc. IEEE Int. Conf. Multimedia Expo. Workshops (ICMEW)*, Jul. 2019, pp. 246–251.

[51] 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. (CVPR)*, Jun. 2020, pp. 2237–2246.

[52] Y. Tan, K. Chang, H. Li, Z. Tang, and T. Qin, "Lightweight color image demosaicking with multi-core feature extraction," in *Proc. IEEE Int. Conf. Vis. Commun. Image Process. (VCIP)*, Dec. 2020, pp. 136–139.

[53] R. Zhou, R. Achanta, and S. Süsstrunk, "Deep residual network for joint demosaicing and super-resolution," in *Proc. Color Imag. Conf.*, vol. 2018, no. 1. Springfield, VA, USA: Society for Imaging Science and Technology, 2018, pp. 75–80.

[54] G. Qian *et al.*, "Rethinking the pipeline of demosaicing, denoising and super-resolution," 2019, *arXiv:1905.02538*.

[55] X. Xu, Y. Ye, and X. Li, "Joint demosaicing and super-resolution (JDSR): Network design and perceptual optimization," *IEEE Trans. Comput. Imag.*, vol. 6, pp. 968–980, Jun. 2020.

[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[57] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," in *Proc. AAAI*, Feb. 2017, pp. 4278–4284.

[58] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 5987–5995.

[59] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.

[60] Z. Gao, L. Guo, W. Guan, A.-A. Liu, T. Ren, and S. Chen, "A pairwise attentive adversarial spatiotemporal network for cross-domain few-shot action recognition-R2," *IEEE Trans. Image Process.*, vol. 30, pp. 767–782, 2021.

[61] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, Jun. 2015, pp. 3431–3440.

[62] X. Li, "Demosaicing by successive approximation," *IEEE Trans. Image Process.*, vol. 14, no. 3, pp. 370–379, Mar. 2005.

[63] X. Li, B. Gunturk, and L. Zhang, "Image demosaicing: A systematic survey," *Proc. SPIE*, vol. 6822, Jan. 2008, Art. no. 68221J.

[64] K. Hirakawa and T. W. Parks, "Adaptive homogeneity-directed demosaicing algorithm," *IEEE Trans. Image Process.*, vol. 14, no. 3, pp. 360–369, Mar. 2005.

[65] D. Zhang and X. Wu, "Color demosaicking via directional linear minimum mean square-error estimation," *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2167–2178, Dec. 2005.

[66] D. Menon, S. Andriani, and G. Calvagno, "Demosaicing with directional filtering and a posteriori decision," *IEEE Trans. Image Process.*, vol. 16, no. 1, pp. 132–141, Jan. 2007.

[67] D. Kiku, Y. Monno, M. Tanaka, and M. Okutomi, "Minimized-Laplacian residual interpolation for color image demosaicking," *Proc. SPIE*, vol. 9023, Mar. 2014, Art. no. 90230L.

[68] W. Ye and K. Ma, "Color image demosaicing using iterative residual interpolation," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5879–5891, Dec. 2015.

[69] Y. Monno, D. Kiku, M. Tanaka, and M. Okutomi, "Adaptive residual interpolation for color image demosaicking," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2015, pp. 3861–3865.

[70] D. Khashabi, S. Nowozin, J. Jancsary, and A. W. Fitzgibbon, "Joint demosaicing and denoising via learned nonparametric random fields," *IEEE Trans. Image Process.*, vol. 23, no. 12, pp. 4968–4981, Dec. 2014.

[71] Z. Ni, K.-K. Ma, H. Zeng, and B. Zhong, "Color image demosaicing using progressive collaborative representation," *IEEE Trans. Image Process.*, vol. 29, pp. 4952–4964, 2020.

[72] X. Zhang, Q. Chen, R. Ng, and V. Koltun, "Zoom to learn, learn to zoom," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3757–3765.

[73] X. Xu, Y. Ma, and W. Sun, "Towards real scene super-resolution with raw images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1723–1731.

[74] C. Chen, Q. Chen, J. Xu, and V. Koltun, "Learning to see in the dark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3291–3300.

[75] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, May 2019, pp. 6105–6114.

[76] G. Liu, A. F. Reda, J. K. Shih, T.-C. Wang, A. Tao, and A. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Munich, Germany, Sep. 2018, pp. 89–105.

[77] J. Cai, S. Gu, and L. Zhang, "Learning a deep single image contrast enhancer from multi-exposure images," *IEEE Trans. Image Process.*, vol. 27, no. 4, pp. 2049–2062, Apr. 2018.

[78] S. Y. Kim, J. Oh, and M. Kim, "Deep SR-ITM: Joint learning of super-resolution and inverse tone-mapping for 4K UHD HDR applications," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3116–3125.

[79] C. Xie, W. Zeng, and X. Lu, "Fast single-image super-resolution via deep network with component learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 12, pp. 3473–3486, Dec. 2019.

[80] J. Pan *et al.*, "Learning dual convolutional neural networks for low-level vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 3070–3079.

[81] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, and F. Durand, "Deep bilateral learning for real-time image enhancement," *ACM Trans. Graph.*, vol. 36, no. 4, p. 118, 2017.

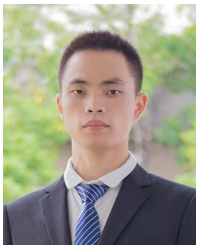
[82] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1122–1131.

[83] R. Timofte, V. D. Smet, and L. V. Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Proc. Asian Conf. Comput. Vis. (ACCV)*, Singapore, Nov. 2014, pp. 111–126.

[84] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2015, pp. 5197–5206.

[85] Y. Matsui *et al.*, "Sketch-based Manga retrieval using manga109 dataset," *Multimedia Tools Appl.*, vol. 76, no. 20, pp. 21811–21838, 2017.
[86] V. Bychkovsky, S. Paris, E. Chan, and F. Durand, "Learning photographic global tonal adjustment with a database of input / output image pairs," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 97–104.
[87] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Color image denoising via sparse 3D collaborative filtering with grouping constraint in luminance-chrominance space," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2007, pp. 313–316.
[88] H. C. Karaimer and M. S. Brown, "A software platform for manipulating the camera imaging pipeline," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Oct. 2016, pp. 429–444.
[89] E. Schwartz, R. Giryes, and A. M. Bronstein, "DeepISP: Toward learning an End-to-End image processing pipeline," *IEEE Trans. Image Process.*, vol. 28, no. 2, pp. 912–923, Feb. 2019.
[90] S. Ratnasingam, "Deep camera: A fully convolutional neural network for image signal processing," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 3868–3878.

Kan Chang received the B.S. degree in communication engineering and the Ph.D. degree in communications and information systems from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2005 and 2010, respectively. From February 2014 to February 2015, he was a Visiting Scholar with the Department of Computer Science, Arizona State University (ASU), Tempe, AZ, USA. He is currently an Associate Professor with the School of Computer and Electronic Information and also a Researcher at the Guangxi Key Laboratory of Multimedia Communications and Network Technology, Guangxi University, Nanning, China. He has authored and coauthored over 50 scientific articles and has obtained ten issued Chinese patents. His research interests include image and video processing, compressive sensing, and video coding.



Hengxin Li received the B.S. degree in communication engineering from Guangxi University, Nanning, China, in 2018, where he is currently pursuing the M.S. degree with the School of Computer and Electronic Information. His research interests include image super-resolution, demosaicking, and denoising.

Yufei Tan received the B.S. degree in electronic engineering from Guangxi Normal University, Guilin, China, in 2016. He is currently pursuing the M.S. degree with the School of Computer and Electronic Information, Guangxi University, Nanning, China. His research interest includes image demosaicking and denoising.

Pak Lun Kevin Ding (Member, IEEE) received the B.S. degree in computing mathematics from the City University of Hong Kong, Hong Kong, in 2013, and the M.A. degree in mathematics from Arizona State University (ASU), Tempe, AZ, USA, in 2015, where he is currently pursuing the Ph.D. degree in computer science. He is a Research Assistant with the Visual Representation and Processing Group. His research interests include machine learning and its applications to computer vision and image processing.

Baoxin Li (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Maryland, College Park, in 2000. From 2000 to 2004, he was a Senior Researcher with SHARP Laboratories of America, Camas, WA, USA, where he was the Technical Lead in developing SHARP's HiIMPACT^{TM} Sports Technologies. He was also an Adjunct Professor with Portland State University from 2003 to 2004. In 2004, he joined Arizona State University (ASU), where he is currently a Professor in computer science and engineering and a Graduate Faculty Endorsed to Chair in the Computer Science, Electrical Engineering, and Computer Engineering Programs. He holds 16 issued U.S. patents. His current research interests include computer vision and pattern recognition, image/video processing, multimedia, medical image processing, and statistical methods in visual computing. He won the SHARP Laboratories' President Award (twice) in 2001 and 2004. He also received the SHARP Laboratories' Inventor of the Year Award in 2002. He received the National Science Foundation's CAREER Award from 2008 to 2009. He is an Associate Editor for IEEE TRANSACTIONS ON IMAGE PROCESSING. Previously, he served as an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and *Signal Processing: Image Communication*.