MDCN: Multi-scale Dense Cross Network for Image Super-Resolution

Juncheng Li, Faming Fang, Jiaqian Li, Kangfu Mei, and Guixu Zhang

Abstract—Convolutional neural networks have been proven to be of great benefit for single-image super-resolution (SISR). However, previous works do not make full use of multi-scale features and ignore the inter-scale correlation between different upampling factors, resulting in sub-optimal performance. Instead of blindly increasing the depth of the network, we are committed to mining image features and learning the inter-scale correlation between different upampling factors. To achieve this, we propose a Multi-scale Dense Cross Network (MDCN), which achieves great performance with fewer parameters and less execution time. MDCN consists of multi-scale dense cross blocks (MDCBs), hierarchical feature distillation block (HFDB), and dynamic reconstruction block (DRB). Among them, MDCB aims to detect multi-scale features and maximize the use of image features flow at different scales, HFDB focuses on adaptively recalibrate channel-wise feature responses to achieve feature distillation, and DRB attempts to reconstruct SR images with different upampling factors in a single model. It is worth noting that all these modules can run independently. It means that these modules can be selectively plugged into any CNN model to improve model performance. Extensive experiments show that MDCN achieves competitive results in SISR, especially in the reconstruction task with multiple upampling factors. The code is provided at https://github.com/MIVRC/MDCN-PyTorch.

Index Terms—Single image super-resolution, multi-scale, feature distillation, dynamic reconstruction.

I. INTRODUCTION

IMAGE super-resolution, especially single image super-resolution (SISR) is an extremely hot topic in the computer vision field, which aims to reconstruct a super-resolution (SR) image from its degraded low-resolution (LR) one. In order to generate high-quality SR images, plenty of SR methods have been proposed, including interpolation-based [1]–[4], anchored neighborhood regression based [5], [6], self-example learning based [7]–[9], and learning based [10], [11] methods. Recently, convolutional neural networks (CNNs) have achieved great success in computer vision tasks, which also profoundly promote the development of SISR. Therefore, CNN-based SR methods [10]–[31] have become the main-stream today, which aim to learn the mapping between LR and HR images by constructing a well-designed network. Among them, Dong et al. [10] proposed the SRCNN, which was the first successful model adopting CNN to the SISR task. After that, Kim et al. [11] introduced residual learning into SISR and proposed the VDSR; Lim et al. [16] introduced residual block and multi-factor strategy to construct EDSR and MDSR models. Li et al. [32] proposed the MSRNR by introducing multi-scale residual learning; Zhang et al. [33] introduced the attention mechanism into residual blocks to construct a very deep RCAN; He et al. [22] presented a cascaded network (CDN_MRF) with multi-receptive fields to increase the spatial resolution; Li et al. [26] proposed a deep adaptive information filtering network (FilterNet) for accurate and fast SR image reconstruction; Apart from these models, more CNN-based SR models can be found in [34] and [35].

Although SR models mentioned above can achieve great results (Fig. 1), they are usually accompanied by complicated structures and huge computational overhead. Among all of them, MSRNR [32] is the focus of this paper. MSRNR [32] was proposed in 2018, which used multi-scale features to reconstruct SR images. To achieve this, the multi-scale residual block (MSRB) was proposed for feature extraction. Now, many works [36]–[45] have proved the effectiveness of multi-scale residual learning strategy and multi-scale features have been used in various fields, such as image denoising [45], remote sensing image super-resolution [41], [42], and medical image enhancement [43], [44]. These models directly migrate
MSRN to other tasks or by optimizing the model structure of MSRN to improve model performance. Although these models greatly improved the architecture and application of MSRN, none of them solved the core problem of MSRN. In this work, we aim to propose a more efficient and general model.

(A). Multi-scale Feature Extraction: Massive studies [46]–[49] have pointed out that image will exhibit different characteristics at different scales and making full use of these features can further improve model performance. Therefore, Li et al. [32] proposed a MSRB for multi-scale feature extraction, which integrated different convolutional kernels in a block to adaptively extract image features at different scales. However, it does not make full use of image features in the previous layers, so local features are difficult to transfer to other layers and the multi-scale feature flow will be suppressed. Therefore, exploring a more efficient multi-scale feature extraction block is essential for image reconstruction.

(B). Hierarchical Feature Distillation: As the depth of the network increases, image features will gradually disappear during the conduction process. Therefore, taking advantage of hierarchical features will greatly improve model performance. However, most models ignore this or simply concatenate all hierarchical features. These methods cannot eliminate redundant features, resulting in sub-optimal results and inefficient image reconstruction. Therefore, an effective method that can exploit hierarchical features and eliminate redundant features is crucial for SISR.

(C). Inter-scale Correlation Exploration: Most SR models introduce the deconvolutional layer or sub-pixel convolutional layer to achieve image magnification. However, due to their characteristics, these models need to train specific models for different upsampling factors. Although some models introduce simple and flexible reconstruction modules, they still fail to achieve the expectation that a model can be suitable for multiple upsampling factors. Therefore, a model that can adapt to multiple factors and learn the inter-scale correlation between different upsampling factors is needed.

To solve these issues, we propose a Multi-scale Dense Cross Network (MDCN). MDCN consists of multi-scale dense cross blocks (MDCBs), hierarchical feature distillation block (HFDB), and dynamic reconstruction block (DRB). Among them, MDCB is an efficient feature extraction module that can extract rich high-frequency details through the integrated dual-path dense network and multi-scale learning. HFDB introduces dimension transformation and channel attention mechanism to adaptively recalibrate channel-wise feature responses, thus redundant hierarchical features will be removed. DRB aims to maximize the reuse of model parameters and learn the inter-scale correlation between different upsampling factors by dynamic activating the corresponding upsampling module. In summary, our contributions are as follows:

(i) We propose a Multi-scale Dense Cross Network (MDCN) for SISR, which achieves competitive results with fewer parameters and less execution time.

(ii) We devise a Multi-scale Dense Cross Block (MDCB) for feature extraction, which is essentially a dual-path dense network that effectively detects local and multi-scale features.

(iii) We design a Hierarchical Feature Distillation Block (HFDB) to maximize the use of hierarchical features. It is the first CNN module specifically designed for hierarchical feature learning.

(iv). We introduce the Dynamic Reconstruction Block (DRB) to learn the inter-scale correlation between different upsampling factors, which enables MDCN to reconstruct SR images with multiple factors in a single model.

II. RELATED WORK

SISR has attracted increasing attention in recent years since the quality of reconstructed SR images will seriously affect the accuracy of high-level computer vision tasks such as image classification [51], [52], image segmentation [53], [54], and object detection [55], [56]. In all relevant studies, feature extraction and multi-factor model have attracted our attention.

Feature Extraction: Recently, numerous feature extraction blocks have been proposed for local feature extraction. In TABLE I, we provide several classic feature extraction blocks, including Residual Block [57], Dense Block [58], Residual Dense Block (RDB [50]), and Multi-scale Residual Block (MSRB [32]). According to the table, we can clearly see the mechanism introduced in each block. Different from previous feature extraction blocks, MSRB introduces a new multi-scale learning strategy. Although this allows MSRB to obtain multi-scale features, it ignores the use of features in the previous layers. In this paper, by rethinking the influence of residual learning, dense connection, and multi-scale learning, we aim to explore a new feature extraction block that can skillfully combine these strategies without adding additional parameters.

On the other hand, as the depth of the network increases, image features will be gradually lost in the process of transmission. Therefore, making full use of hierarchical features is also important for image restoration. In Fig. 2, we show four...
different methods, including the series connection (A, without using hierarchical feature), residual connection (B, only use the input image features), dense residual connection (C, use all hierarchical features for residual learning), and hierarchical connection (D, concatenate all hierarchical features and apply a $1 \times 1$ layer for feature fusion). However, by investigating the performance of methods A-D, we find that hierarchical features can improve the model performance while redundant hierarchical features will make the model more difficult to train. Therefore, a new method that can effectively utilize hierarchical features and eliminate redundant features is crucial.

Multi-factor Model: A model that can be applied to multiple upsampling factors is the hot topic in SISR, which is the basic condition for inter-scale correlation learning. Nonetheless, due to the limitations of deconvolutional layer and sub-pixel convolutional layer, there is still no perfect solution for this problem. Recently, some models have been proposed for multiple upsampling factors, such as LapSRN [15] and MDSR [16]. LapSRN [15] (Fig. 3 (A)) is a Laplacian pyramid framework, which can progressively reconstruct high-resolution images. However, this method essentially decomposes the large upsampling factor into multiple small upsampling factors for image reconstruction, which does not consider the inter-scale correlation between different upsampling factors. MDSR [16] (Fig. 3 (B)) introduces the scale-specific processing modules at the head and tail of the model to handle different upsampling factors. This structure can be suitable for multiple upsampling factors, but the scale-specific processing modules at the head of the model are not conducive to the transmission of information and the interaction of inter-scale features. In order to maximize the reuse of model parameters and further exploit the inter-scale correlation between different upsampling factors, we aim to further optimize the strategy used in MDSR and propose a more efficient method.

III. Multi-scale Dense Cross Network (MDCN)

In this paper, we propose a Multi-scale Dense Cross Network (MDCN) for SISR. As shown in Fig. 4, MDCN can be divided into two stages: feature extraction and dynamic reconstruction. In stage I, we use $N$ multi-scale dense cross blocks (MDCBs) and a hierarchical feature distillation block (HFDB) for local feature extraction and hierarchical features distillation, respectively. In stage II, we introduce a dynamic reconstruction block (DRB) for SR image reconstruction. This block allows the model to be suitable for multiple upsampling factors, which makes MDCN more scalable.

Define $I_{LR}$ and $I_{SR}$ as the input and output of MDCN. $L_{input}$, $L_{output}$, and $L_{mix}$ are the input of the first MDCB, the output of the last MDCB, and the input of DRB, respectively. Following previous works, we first use a $3 \times 3$ convolutional layer to upgrade the LR image to a high dimensional

$$L_{input} = F_{input}(I_{LR}),$$

where $F_{input}(\cdot)$ denotes the corresponding convolutional layer and $L_{input}$ is the converted features. Meanwhile, $L_{input}$ is also served as the input of MDCB for local feature extraction

$$L_{output} = F_{MDCG}(L_{input}),$$

where $F_{MDCG}(\cdot)$ represents the multi-scale dense cross group (MDCG), which consists of $N$ MDCBs. To make full use of the hierarchical features, we also introduce the HFDB for hierarchical feature fusion and distillation

$$L_{dis} = F_{HFDB}([L_{bic}, L_{bic}^{2}, \ldots, L_{bic}^{N-1}]),$$

where $F_{HFDB}(\cdot)$ denotes the HFDB and $L_{bic}^{n}$ ($n = 1, 2, \ldots, N - 1$) represents the output of the $n$-th MDCB. In addition, $L_{dis}$ denotes the distilled hierarchical features, which can be used for image reconstruction.

After feature extraction, we combine the original image feature $L_{input}$, the extracted high-level features $L_{output}$, and the distilled hierarchical features $L_{dis}$ for the final SR image reconstruction

$$L_{mix} = F_{mix}(L_{input} + L_{dis} + L_{output}),$$

where $F_{mix}(\cdot)$ is a $3 \times 3$ convolutional layer used for feature fusion. Finally, the merged features are delivered to the DRB for high-quality SR image reconstruction and a $3 \times 3$ convolutional layer is applied to convert it to RGB space

$$I_{SR} = F_{DRB}(L_{mix}),$$

where $F_{DRB}(\cdot)$ denotes the dynamic reconstruction block, $F_{output}(\cdot)$ represents the corresponding operation that converts image feature maps from high dimensional space to RGB space, and $I_{SR}$ is the finally reconstructed SR image.

During training, MDCN is optimized with L1 loss function. Therefore, Given a training dataset $\{I_{LR}, I_{HR}\}_{i=1}^{M}$, we solve

$$\hat{\theta} = \arg \min_{\theta} \frac{1}{M} \sum_{i=1}^{M} \left\| F_{\theta}(I_{LR}) - I_{HR} \right\|_1,$$

where $\theta$ denotes the parameter set of our model and $F(\cdot)$ denotes the MDCN. Each module of MDCN will be described in the following section.
A. Multi-scale Dense Cross Block (MDCB)

Features extraction is the most important step in image restoration. In order to make full use of image features from the LR image, we propose a new module named multi-scale dense cross block (MDCB). MDCB is the basic component of MDCN, which is inspired by MSRB [32] and aims to solve the problem that MSRN can not obtain features from the previous layers. Compared with MSRB [32], MDCB adopts a dual dense network as the backbone, which enables it to extract image features at different scales as well as take advantage of features from the previous layers.

Dual-path Network: As shown in Fig. 5, MDCB is essentially a dual-path network, which uses two dense networks as the backbone. In order to better explain the working mechanism of the module, we provide the decomposed structure of MDCB in Fig. 6. In Fig. 6, (A) is the simplified version of MDCB, which removes residual learning for better explanation, (B) is the decomposed structure of MDCB, and (C) is the equivalent structure after straightening (B). As shown in Fig. 6 (C), the model can be divided into two parts: DenseNet-Top and DenseNet-Bottom. Each part is actually a modified dense network [58], which reduces the depth of the network and introduces bottleneck layers for features fusion. Following the dense network, our Dense-Top or DenseNet-Bottom connects each layer to every other layers in a feed-forward fashion (red lines), which can detect rich features in the previous layers. The operations of Dense-Top are:

\[
L_{22} = C_{3 \times 3}^1(L_{11}),
\]

\[
L_{33} = C_{3 \times 3}^2(C_{1 \times 1}^1([L_{12}, L_{22}])),
\]

\[
L_{\text{out}} = C_{1 \times 1}^3([L_{13}, L_{23}, L_{33}]),
\]

where \(C_{p \times q}^r\) denotes the \(p\)-depth \(3 \times 3\) convolutional layer, \(C_{1 \times 1}^p\) denotes the \(p\)-depth \(1 \times 1\) convolutional layer, and \([\cdot, \cdot, \cdot]\) denotes the concatenation operation. In this part, we use \(3 \times 3\) convolutional layers for feature extraction and use \(1 \times 1\) convolutional layer for feature fusion. Inspired by MSRB [32], we also introduce multi-scale learning to extract image features at different scales. Therefore, we replace all \(3 \times 3\) convolutional layers with \(5 \times 5\) convolutional layers in the DenseNet-Bottom

\[
H_{22} = C_{5 \times 5}^1(H_{11}),
\]

\[
H_{33} = C_{5 \times 5}^2(C_{1 \times 1}^2([H_{12}, H_{22}])),
\]

\[
H_{\text{out}} = C_{1 \times 1}^3([H_{13}, H_{23}, H_{33}]).
\]

Feature Exchange & Fusion Mechanism: As mentioned above, MDCB uses two dense networks to extract image features at different scales. However, if these two subnets are independent of each other, image features at different scales will be difficult to transfer and fuse. Therefore, we associate the DenseNet-Top and DenseNet-Bottom by introducing two skip connections: \(M_{35}\) and \(M_{53}\). As shown in Fig. 6, the blue lines are the introduced skip connection. \(M_{35}\) and \(M_{53}\) transmit the unique features extracted by themselves to the other sub-net. This facilitates the exchange of image features at different scales and improves the model performance. Meanwhile, we also introduce a bottleneck layer at the tail of the module to achieve feature fusion and dimensionality reduction, which combines the multi-scale features (\(L_{33}\) and \(H_{33}\)) extracted by DenseNet-Top and DenseNet-Bottom with the local features (\(L_{23}, H_{23}\), and \(L_{13}/H_{13}\)) provided by the previous layers to obtain the high-frequency features. Therefore, the complete operations of multi-scale dense feature extraction can be defined as

\[
L_{22} = C_{3 \times 3}^1(L_{11}),
\]

\[
H_{22} = C_{5 \times 5}^1(H_{11}),
\]

\[
L_{33} = C_{3 \times 3}^2(C_{1 \times 1}^1([L_{12}, L_{22}, M_{23}])),
\]

\[
H_{33} = C_{5 \times 5}^2(C_{1 \times 1}^2([H_{12}, H_{22}, M_{33}])),
\]

\[
L_{\text{out}} = C_{1 \times 1}^3([L_{23}, L_{33}, H_{23}, H_{33}, L_{\text{in}}]).
\]

Among them, \(L_{\text{in}} = L_{11} = L_{12} = L_{13} = H_{11} = H_{12} = H_{13}, L_{22} = L_{23} = M_{55}\), and \(H_{22} = H_{23} = M_{53}\), which represent
the same features while delivering to different convolutional layers. In addition, $L_{in}$ and $L_{out}$ denote the input and output of MDCB, respectively.

**Local residual learning:** Following previous works [32], [50], we also introduce local residual learning into our MDCB to further improve the information flow. Therefore, the output of $n$-th MDCB can be written as $L_n = L_{n-1} + L_{out}$. $L_{n-1}$ denotes the output of the previous MDCB and serves as the input of this MDCB ($L_{in} = L_{n-1}$).

### B. Hierarchical Feature Distillation Block (HFDB)

As mentioned in MSRN [32], hierarchical features can further improve model performance. However, the method used in MSRN is crude. This is because only use the $1 \times 1$ convolutional layer to compress hierarchical features can not extract effective features. Meanwhile, this method will produce massive redundant features, which is not conducive to model training. In order to remove redundant features and fully mine useful hierarchical features, we propose a hierarchical feature distillation block (HFDB, Fig. 7). HFDB focuses on adaptively recalibrate channel-wise feature response to achieve feature distillation, which is an efficient module that only needs low computational overhead. The core of HFDB is the introduced dimension transformation and channel attention mechanism.

**Dimension Transformation Mechanism (DTM):** Inspired by AutoEncoder, we introduce the dimension transformation mechanism to the module to achieve feature distillation. As shown in Fig. 7, HFDB introduces the bottleneck layer ($1 \times 1$ layer) at the head and tail of the module, respectively. Firstly, all hierarchical features are concatenate like MSRN [32]. Then, these $M$ feature maps are sent to the bottleneck layer to reduce the feature dimension. Then, the compressed features are sent to the channel attention module to adaptively recalibrate channel-wise feature response. Finally, the dimension of recalibrated features will be upgraded by the bottleneck layer at the tail of the module. It is worth noting that we set $M > C > 96$. Therefore, this module achieves a similar effect to the AutoEncoder, thus achieving feature distillation.

**Channel Attention Mechanism (CAM):** As mentioned above, we introduce CAM to the module to mine the most useful hierarchical features. CAM was firstly proposed in SENet [59], which could adaptively calibrate channel-wise feature response by explicitly modeling interdependency between channels. Inspired by this, RCAN [33] and SrSENet [60] introduced CAM into all feature extraction blocks to obtain useful local features. However, this method will only bring a slight performance improvement while the increasement in execution time and memory consumption is huge [28]. Different from RCAN [33] and SrSENet [60] that pay attention to the local features extraction, we focus on using CAM to mine the relationship between hierarchical features and extract the most useful hierarchical features.

Taking $X = [x_1, \ldots, x_C]$ as input, which has $C$ feature maps with size of $H \times W$. The $c$-th element of $z$ can be defined as

$$z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} x_c(i, j),$$

where $x_c(i, j)$ is the value at position $(i, j)$ of $c$-th feature $x_c$. Then, we apply a gating mechanism with the sigmoid function $s = f(C_1(\delta(C_2(z))))$, where $f(\cdot)$ and $\delta(\cdot)$ denote the sigmoid function and ReLU function, respectively. $C_1(\cdot)$ is a convolutional layer, which acts as channel-downscaling with reduction ratio $r$ and is activated by ReLU, the low-dimensional feature is increased with ratio $r$ by a channel-upscaling layer $C_2(\cdot)$. Finally, the channel statistics $s$ is used to rescale the input: $x'_c = s_c \cdot x_c$, where $s_c$ and $x_c$ are the scaling factor and feature maps of the $c$-th channel.

Due to the ability of DTM and CAM, HFDB can not only eliminate redundant features but also extract useful hierarchical features. Meanwhile, MDCN only uses the channel attention mechanism once, which can greatly improve the efficiency of the model.
C. Dynamic Reconstruction Block (DRB)

In order to exploit the inter-scale correlation between different upsampling factors and maximize the use of model parameters, we introduce the dynamic reconstruction block (DRB). DRB is essentially a multi-element module that combines several different upsampling modules, which was first proposed in MDSR [16]. For single upsampling factor, DRB will create a corresponding scale-specific upsampling module. However, as for multi-factor image super-resolution task, DRB will generate a set of corresponding scale-specific upsampling modules (e.g., ×2, ×3, and ×4). In the upsampling module, we use the sub-pixel convolutional layer [14] to learn an array A. Implementation Details

The performance and generalization capability of each model.

benchmark datasets, which can provide a fair comparison of our test datasets. All of them are the most widely used (e.g., ×2, ×3, and ×4 upsampling modules). Meanwhile, these upsampling modules are arranged in parallel and only one upsampling module will be activated when used. In other words, the data stream will only flow into the module corresponding to the upsampling factor for image reconstruction. To better exploit the inter-scale correlation between different upsampling factors, we construct a dataset with different factors and introduce a mixed training strategy for training. This means that different size of images will be randomly sent to the model. Then it will automatically activate the corresponding upsampling modules according to the upsampling factors. Compared with specifically training a model for each upsampling factor, this method can learn the latent inter-scale information, maximize the reuse of model parameters, and reduce the training time. Thanks to this strategy, the trained model can be directly applied to multiple upsampling factors without re-training.

Different from MDSR [16], all parameters of our model are shared under different upsampling factors, except for the DRB. This can maximize the reuse of model parameters and benefit for the model to learn the inter-scale correlation between different upsampling factors, thus improve the model performance and robustness. In Sec. V-C, we will further verify the effectiveness of this module.

IV. EXPERIMENTS

Following previous works, we only use DIV2K (1-800) [61] as our training dataset. Meanwhile, we choose Set5 [62], Set14 [63], B100 [64], Urban100 [9], and Manga109 [65] as our test datasets. All of them are the most widely used benchmark datasets, which can provide a fair comparison of the performance and generalization capability of each model.

A. Implementation Details

Model setting: In the final version of MDCN, we use 12 MDCBs (N = 12) for feature extraction, the input and output channel of each MDCB are set as 128 (C = 128). Detailed configuration of the input and output channels for each layer in each module has been provided in Fig. 5 and Fig. 7. We also introduce the self-ensemble mechanism [68] to further improve MDCN, which is denoted as MDCN+. Specifically, we first flip and rotate the input image to generate seven augmented inputs for each sample, thereby, we can obtain eight corresponding inputs. Then, we reconstruct the corresponding SR images using our MDCN. Finally, we apply the inverse transformation to those SR images and average them to generate the final SR image.

Training setting: During training, we use RGB image as input and augment the image with random horizontal flips and rotations. It is worth noting that our MDCN is a multi-factor model. Therefore, we use the mix training method to train the model. During training, we extract 16 LR patches with a randomly selected scale among ×2, ×3, and ×4, 1,000 iterations of back-propagation constitute an epoch. The learning rate is initialized to 10−4 and halved every 200 epochs. Different from previous works [32], [33] which set the size of LR images as 48×48, we set the size of HR images as 48×48. Hence, the size of the corresponding LR images of different upsampling factors is 24×24, 16×16 and 12×12, respectively. Using small patch size for training will cause performance degradation, but greatly reduce the training time. However, our MDCN still achieves great results due to the fully use of the inter-scale correlation between different upsampling factors. We implement our model with the PyTorch framework and update it with Adam optimizer, all our experiments are performed on GTX TitanX.

B. Comparisons with state-of-the-art SR methods

We compare MDCN with more than 19 SR methods, including Bicubic, A+ [6], SelfExSR [9], SRCNN [10], ESPCN [14], FSRCNN [12], VDSR [11], LapSRN [15], DRCN [13], MRFN [27], SRMDNF [21], MSRN [32], EDSR [16], RDN [50], RCAN [33], FilterNet [26], DNCL [30], RAN [29], and SeaNet [31]. All SR images are evaluated with PSNR and SSIM [69] on the Y channel in YCbCr space.

Quantitative Comparison: In TABLE II, we show the quantitative comparisons with some advanced SR methods, all of them have achieved competitive results at the time. Among them, best results are highlighted and the second best results are underlined. Besides, the ‘Average’ denotes the average results of these 5 test datasets. Obviously, our MDCN achieves competitive results on all upsampling factors. Among them, RDN is slightly better than MDCN under small upsampling factors (×2). However, our MDCN can achieve better results under large upsampling factors (e.g., ×3, ×4). This is because the introduced MDCB in MDCN can fully extract multi-scale features, which is conducive to large upsampling factor SR reconstruction. Considering that RCAN achieves the state-of-the-art results, we make a detailed comparison with it in TABLE III. We can observe that RCAN is slightly better than MDCN (×2: 0.24dB, ×3: 0.15dB, and ×4: 0.09dB). However, it should be noticed that the execution time of our MDCN is 3 times faster than RCAN. This means that MDCN can achieve similar results as RCAN with less execution time. Meanwhile, it is worth noting that: (1). Except for MDCN, all reported SR methods are specially trained for different upsampling factor; (2). EDSR [16], RDN [50], and RCAN [33] use the pre-trained model (×2) as the initialization model to train large upsampling factor model like ×4; (3). DRCN [13] introduces the recursive mechanism to further improve model
TABLE II
QUANTITATIVE COMPARISONS WITH STATE-OF-THE-ART SR METHODS. THE BEST RESULTS ARE HIGHLIGHTED AND THE SECOND BEST RESULTS ARE UNDERLINED. OBVIOUSLY, MDCN ACHIEVES COMPETITIVE RESULTS ON ALL BENCHMARK DATASETS.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
</tr>
<tr>
<td>SRDNN (2018)</td>
<td>x2</td>
<td>33.66 / 0.9299</td>
<td>30.24 / 0.8688</td>
<td>29.56 / 0.8431</td>
<td>26.88 / 0.8403</td>
<td>30.80 / 0.9399</td>
<td>30.22 / 0.8832</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
</tr>
<tr>
<td></td>
<td>x2</td>
<td>33.66 / 0.9299</td>
<td>30.24 / 0.8688</td>
<td>29.56 / 0.8431</td>
<td>26.88 / 0.8403</td>
<td>30.80 / 0.9399</td>
<td>30.22 / 0.8832</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MDCN (Ours)</td>
<td>x2</td>
<td>33.66 / 0.9299</td>
<td>30.24 / 0.8688</td>
<td>29.56 / 0.8431</td>
<td>26.88 / 0.8403</td>
<td>30.80 / 0.9399</td>
<td>30.22 / 0.8832</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
</tr>
<tr>
<td></td>
<td>x2</td>
<td>33.66 / 0.9299</td>
<td>30.24 / 0.8688</td>
<td>29.56 / 0.8431</td>
<td>26.88 / 0.8403</td>
<td>30.80 / 0.9399</td>
<td>30.22 / 0.8832</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MDCN+ (Ours)</td>
<td>x2</td>
<td>33.66 / 0.9299</td>
<td>30.24 / 0.8688</td>
<td>29.56 / 0.8431</td>
<td>26.88 / 0.8403</td>
<td>30.80 / 0.9399</td>
<td>30.22 / 0.8832</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

 TABLE III
QUANTITATIVE COMPARISONS (PSNR/SSIM, PARAMETERS, AND EXECUTION TIME) WITH RCAN [33]. THE BEST RESULTS ARE HIGHLIGHTED AND THE FASTEST EXECUTION TIME ARE RED.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Scale</th>
<th>Parameters</th>
<th>PSNR / SSIM</th>
<th>PSNR / SSIM</th>
<th>PSNR / SSIM</th>
<th>PSNR / SSIM</th>
<th>PSNR / SSIM</th>
<th>PSNR / SSIM</th>
<th>PSNR / SSIM</th>
<th>PSNR / SSIM</th>
<th>PSNR / SSIM</th>
<th>PSNR / SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
</tr>
<tr>
<td></td>
<td>x2</td>
<td>33.66 / 0.9299</td>
<td>30.24 / 0.8688</td>
<td>29.56 / 0.8431</td>
<td>26.88 / 0.8403</td>
<td>30.80 / 0.9399</td>
<td>30.22 / 0.8832</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MDCN (Ours)</td>
<td>x2</td>
<td>33.66 / 0.9299</td>
<td>30.24 / 0.8688</td>
<td>29.56 / 0.8431</td>
<td>26.88 / 0.8403</td>
<td>30.80 / 0.9399</td>
<td>30.22 / 0.8832</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
<td>PSNR / SSIM</td>
</tr>
<tr>
<td></td>
<td>x2</td>
<td>33.66 / 0.9299</td>
<td>30.24 / 0.8688</td>
<td>29.56 / 0.8431</td>
<td>26.88 / 0.8403</td>
<td>30.80 / 0.9399</td>
<td>30.22 / 0.8832</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MDCN+ (Ours)</td>
<td>x2</td>
<td>33.66 / 0.9299</td>
<td>30.24 / 0.8688</td>
<td>29.56 / 0.8431</td>
<td>26.88 / 0.8403</td>
<td>30.80 / 0.9399</td>
<td>30.22 / 0.8832</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
plenty of previous works have demonstrated their effectiveness. Other models use large LR images as inputs for training. All these strategies can further boost the performance. However, in order to verify the effectiveness of MDCN, we do not use any training tricks in our experiment. Nevertheless, since MDCB, HFDB, and DRB can extract rich image features and learn the inter-scale correlation, our MDCN still achieves competitive results.

**Visual Comparison:** In Figs. 8 and 9, we show visual comparisons under small (×2, ×3) and large upsampling factor (×4), respectively. Among them, EDSR [16] was the champion model of the NTIRE2017 SR Challenge, RDN [50] and RCAN [33] are superior models that achieved SOTA results. According to the figure, we can clearly observe that: (i) Most compared SR methods (e.g., SRCNN, MSRN, and SeaNet) cannot recover clear and accurate image edges. Furthermore, under large upsampling factor (e.g., ×4), the reconstructed SR images are blurred with severe artifacts and incorrect edges. In contrast, our MDCN can reconstruct more realistic SR images with clear and sharp edges; (ii) Compared with large size models (e.g., EDSR, RDN, and RCAN), our MDCN shows competitive performance and better results in edges reconstruction. Overall, with the help of MDCB and HFDB, MDCN can reconstruct high-quality SR images.

**V. Model Analysis**

A. **Study of Multi-scale Dense Cross Block (MDCB)**

MDCB is the most important component of MDCN, which is designed for local and multi-scale feature extraction.

(1) As mentioned in Sec. III-A, MDCB is essentially a dual-path dense network, which introduces multi-scale learning, feature exchange & fusion mechanism, and residual learning. In TABLE IV, we provide a series of ablation studies to investigate their effectiveness.

**Table IV**

<table>
<thead>
<tr>
<th>Case Index</th>
<th>MDCB</th>
<th>FEFM</th>
<th>RL</th>
<th>HFDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Dual-path Dense Network:** In Cases 1 and 2, we remove all introduced mechanisms, only leaving DenseNet-Top or DenseNet-Bottom, respectively. In Case 3, we use both DenseNet-Top and DenseNet-Bottom to build the dual-path network for image reconstruction. It is worth noting that Case 3 is a simple combination of these two dense networks and does not introduce skip connections (M35 and M53) for feature exchange and fusion. According to the table, we can clearly see that Case 3 achieves better results, which demonstrates the dual-path network is effective.

**Feature Exchange & Fusion Mechanism (FEFM):** We introduce skip connections (M35 and M53) in Case 4 to achieve feature exchange and fusion. Compared with Case 3, Case 4 achieves better results. This is because the introduced FEFM can transfer the different scales of image features to the other part, which greatly enriches the diversity of extracted features. Therefore, the model can achieve better results.

**Residual Learning (RL):** Plenty of previous works have proved the effectiveness of residual learning. Following these works, we also introduce residual learning into our MDCN to improve the information flow. In Cases 4 and 5, we provide
used feature extraction blocks, which have well-designed
RCAB [33], and MSRB [32]. All of them are the most widely
we compare our MDCB with ResBlock [57], RDB [50],
N
model size, we set
in order to achieve a well balance between performance and
and its performance still has room for improvement. However,
channel increases, the model performance can be further
Cases 6, 7, and 8, we provide the results of different numbers
network, so the size of the model can be easily adjusted by
convergence and solve the problem of gradient disappearance
improved; (iii) the performance of MDCB is comparable to
and with HFDB, respectively. Obviously, with the help of
HFDB, the model performance can be further improved. On
the other hand, we compare HFDB with four hierarchical
features, we propose HFDB. In TABLE IV, Cases 5 and 6 represent the results of the model without and the performance curves of each method are
same model and the performance curves of each method are
fairness of comparison, we apply these methods to the
retrained and tested under the same equipment, environment,
and dataset. Meanwhile, all of these feature extraction blocks
are adjusted to the same parameter level by adjusting the
number of blocks or channels for a fair comparison. In
Fig. 10, we show the performance curve of these models
during training. According to this figure, we can draw the
following conclusions: (i) simply stacking ResBlock or RCAB
will not effectively boost model performance; (ii) compared
with MSRB, the result of MDCB has been significantly
improved; (iii) the performance of MDCB is comparable to
RDB, even slightly better than it. This is because MDCB
skillfully combines multi-scale residual learning with dense
can not effectively boost model performance; (ii) compared
with MSRB, the result of MDCB has been significantly
improved; (iii) the performance of MDCB is comparable to
RDB, even slightly better than it. This is because MDCB
skillfully combines multi-scale residual learning with dense
connections to make it have the characteristics of both MSRB
and RDB, so as to obtain better results. All aforementioned
experiments fully demonstrate the effectiveness of MDCB.

**B. Study of Hierarchical Feature Distillation Block (HFDB)**

In order to eliminate redundant features and extract the most
useful hierarchical features, we propose HFDB. In TABLE IV,
Cases 5 and 6 represent the results of the model without
and with HFDB, respectively. Obviously, with the help of
HFDB, the model performance can be further improved. On
the other hand, we compare HFDB with four hierarchical
feature utilization methods mentioned in Fig. 2. To ensure
the fairness of comparison, we apply these methods to the
same model and the performance curves of each method are
presented in Fig. 11. Obviously, our HFDB achieves the best
results (the blue line). It is worth noting that the introduced
dimension transformation mechanism in HFDB can greatly

![Fig. 9. Visual comparison with different SR methods on Urban100 [67] under large upsampling factor (×4). Please zoom in to view details.](image)

![Fig. 10. Performance comparison between different feature extraction blocks.](image)
reduce model parameters. Taking 12 MDCBs as an example, Method D needs 245,760 parameters while our HFDB only needs 173,184 parameters, \( \frac{5}{7} \) of Method D. In addition, this gap will increase as the number of MDCB increases. In summary, compared with Method D (proposed in MSRN [32]), our HFDB can achieve better results with fewer parameters and less computational overhead.

C. Study of Dynamic Reconstruction Block (DRB)

To realize the reconstruction of SR images with different upsampling factors in a single model, we introduce DRB into MDCN. DRB can create a set of scale-specific upsampling modules according to actual needs. This can maximize the reuse of model parameters and enables the model to learn the inter-scale correlation between different upsampling factors. It is worth noting that when there is only one upsampling module in DRB, it is the same as previous works that specifically trained for different upsampling factors. This special case is named as 'MDCN-S'. In order to illustrate the effectiveness of DRB, we provide the following experiments:

1. In TABLE V, we provide PSNR results of MSRN, MDCN-S, and MDCN on 5 benchmark test datasets. According to the table, we can observe that: (a) whether MDCN-S or MDCN, the performance is better than MSRN; (b) compared with MDCN-S, the performance of MDCN has been significantly improved. This proves that the inter-scale correlation can further improve the model performance.

2. As described in Section II, DRB was first proposed in MDSR [16]. Different from MDSR [16], we only apply DRB at the tail of the model. This means that all feature extraction modules in MDCN are weight sharing, which benefits for inter-scale learning. In TABLE VI, we show the quantitative comparisons of multi-factor models, including VDSR [11], MDSR [16], and our MDCN. For a fair comparison, all models are re-trained under the same training dataset and settings. Among them, \((\times 2)\), \((\times 3)\), and \((\times 4)\) stand for the specifically trained model for one single upsampling factor and \((\times 2,\times 3,\times 4)\) denote the multi-factor model with mix training method. According to the table, we can find that: (i). Our MDCN achieves the best results under all upsampling factors whether using the single factor or multi-factor mixed training method; (ii). Most results of multi-factor models (mixed training) are superior to the special training models (single factor training). This indicates that fully using the inter-scale correlation between different upsampling factors can improve the model performance; (iii). At the small upsampling factor \((\times 2)\), the results of single-factor MDCN are slightly better than the mixed trained MDCN. This is because SR images reconstructed with small upsampling factor is a relatively simple task, so that the introduced other scale information will interference it. Fortunately, this gap is acceptable. Therefore, we recommend to introduce DRB and mix training strategy to build a general model.

VI. RESEARCH AND ANALYSIS

A. Compare with MSRN

Different from MSRN, MDCN introduced MDCB, HFDB, and DRB to build a more efficient and general SR model. The effectiveness of these modules have been verified in Sec.V. Considering the similarity of these models, we provide more detailed comparisons to manifest the effectiveness of MDCN.

1. In TABLE II, we provide the PSNR and SSIM results of MSRN and MDCN. Obviously, MDCN achieves better results. In addition, we provide some high-frequency detail comparisons between MSRN and MDCN in Fig. 12. We can clearly observe that the high-frequency details of the
image reconstructed by MSRN are severely damaged, and even wrong textures and edges are generated. Contrastly, our MDCN can reconstruct high-quality SR images with accurate and clear details. Meanwhile, different from MSRN that specially trained for various upsample factors, we only use one MDCN to perform SR reconstruction task for different factors.

(2) Considering that the parameter amounts of MSRN and MDCN are not equal, we design two experiments to further explore their performance. To achieve this, we build two models, named MDCN’ and MSRN’, respectively. MDCN’ is a variant of MSRN, which uses MSRN as the backbone and replaces all MSRBs in the network with MDCB ($C = 64$). In contrast, MSRN’ is the variant of MSRN, which uses MSRN as the backbone and replaces all MSRBs in the network with MDCB ($C = 128$). Quantitative comparisons are shown in TABLE VII and VIII, respectively. Compared with MSRN, MDCN’ achieves better results on all benchmark test datasets with fewer parameters. This also verifies the feature extraction ability of MDCB is superior to MSRB. Similarly, MDCN achieves better results than MSRN’ with fewer parameters. This fully proves the effectiveness of MDCN.

In summary, although MSRN is an efficient SR model, the performance can be further improved. Meanwhile, extensive experiments demonstrate that MDCN can achieve better results than MSRN with fewer parameters.

### B. Compare with Other Multi-scale SR Models

As described in Sec. I, many improved versions of MSRN have been proposed such as MSDN [70], MSFFRN [38], MWRN [71], and MSRCAN [36]. They introduce dense connection, feature fusion, wide-activated, or channel attention mechanisms to improve the performance of MSRN. However, these models do not pay attention to the core defects of MSRN, so the introduced mechanism does not significantly boost model performance. To solve this problem, we proposed MDCN. As shown in TABLE IX, we provide the quantitative comparisons of these models. According to the table, we can clearly observe that the performance of MDCN has been greatly improved compared to others. This further demonstrates the effectiveness of MDCN.

### C. Investigation of Image Naturalness

In Sec. IV, we evaluate the performance of MDCN from PSNR, SSIM, and visual effects. In order to further verify the distribution and perceptual quality of the reconstructed images, we introduce new indicators in this part, including Root Mean Square Error (RMSE) and Perceptual Index [72] (PI). Among them, RMSE is used to measure the standard deviation of the difference between SR and HR images, and PI is a new criterion that bridges the visual effect with computable index. Specifically, PI is a perceptual quality index which is judged by the $Ma$ and NIQE scores:

$$PI = \frac{1}{2}((10 - Ma) + NIQE),$$

where $Ma$ is the non-reference measures of Ma’s [73] score and NIQE [74] is a natural image quality evaluator. Meanwhile, a lower perceptual index (PI) represents a better perceptual quality.

In Fig. 13, we provide the results of RMSE and PI. According to the top figure, we can clearly see that the RMSE result of MDCN is much lower than other methods. This means that the SR images reconstructed by MDCN have a closer data distribution to real HR images, in other words, the reconstructed images have higher quality. For the below one, we can observe that the PI results of EDSR [16] and MDCN are smaller than others and the performance of them
are very close. This indicates that the images reconstructed by EDSR and MDCN have better perceptual quality and image naturalness. All these results fully manifest that our MDCN can achieve competitive performance.

### E. Model Size and Execution Time

Increasing the depth of the network is the easiest way to improve model performance. Therefore, some large size models have been proposed in recent years such as EDSR [16] and RDN [50]. However, it cannot be ignored that these models are also accompanied by numerous parameters. It means that these models require more storage space, computing resources, and execution time. In Fig. 15, we show the comparison of model parameters and execution time. Among them, red dots represent our MDCN', MDCN, and MDCN+, respectively. According to the figure, we can draw the following conclusions: (i) Compared with lightweight SR models (e.g., SRCNN, VDSR, and MSRN), the performance of MDCN is greatly improved; (ii) Compared with large models (e.g., EDSR and RDN), MDCN achieves close or better results with fewer parameters and less execution time; (iii) Compared with RCAN, we can find that the performance of RCAN is slightly better than MDCN. However, it should not be ignored that the execution time of RCAN is 3 times of MDCN. Furthermore, MDCN can be suitable for multiple upsampling factors (x2, x3, and x4) without any re-training. This means that MDCN can save lots of storage space, training time, and computational overhead. In summary, MDCN achieves a well balance between model performance, model size, and execution time.

### VII. DISCUSSION

Although MDCN is an efficient and general SR model, it still has some limitations:

1. In this paper, we focus on the reconstruction effect of MDCN on simulated degradation (e.g., Bicubic down-sampling) images and verify its effectiveness from multiple indicators (e.g., PSNR, SSIM, RMSE, PI, and visual effect). It is worth noting that the task of real image super-resolution is more difficult due to the unknown and random degradation modes. However, this does not mean that our research is meaningless. Like previous works [10]–[21], [25]–[32], we
concentrate on the design of efficient and universal network structure. Recently, some methods [77]–[80] have been proposed for real image super-resolution. These models use real-world SR datasets for training, thus the model can achieve real image super-resolution. This manifests that MDCN can also be implemented to real image super-resolution by fine-tuning on the real SR dataset. We will explore the performance of MDCN on real images in future works.

(2) To make MDCN applicable to multiple upsampling factors without any re-training, we introduce the dynamic reconstruction block (DRB) to learn the inter-scale correlation between different upsampling factors. However, due to the limitations of sub-pixel convolutional layer, this method cannot directly handle non-integer factors. In order to remedy this problem, we suggest combining our MDCN with Bicubic to handle any upsampling factors, including non-integer factors. Specifically, we first utilize MDCN to perform integer magnification, and then use Bicubic to adjust the non-integer part. For example, for the factor of $\times 3.2$, use MDCN to process $\times 3$ first, then apply Bicubic to adjust it to the $\times 3.2$. Although the strategy is simple, it is very effective. At the same time, the performance of this strategy far exceeds the method of using Bicubic to directly enlarge the image. We also notice that some researchers [81], [82] claim that their models can deal with arbitrary upsampling factors. In the future work, we will further explore the reliability of these strategies and improve our MDCN to handle arbitrary upsampling factors.

VIII. CONCLUSIONS

In this paper, we propose a Multi-scale Dense Cross Network (MDCN) for SISR. MDCN is a robust SR model that can be applied to multiple upsampling factors without any re-training. Specifically, the proposed multi-scale dense cross blocks (MDCB), hierarchical feature block (HFDB), and dynamic reconstruction block (DRB) together form the MDCN. Among them, MDCB aims to fully detect and utilize local and multi-scale features. HFDB focuses on maximizing distillation and uses hierarchical features, and the introduced DRB enables the model to learn inter-scale correlations between different upsampling factors. Extensive evaluations demonstrate that MDCN can achieve competitive results with fewer parameters and less execution time, which achieves an excellent balance between model size and performance. Furthermore, MDCN can be easily expanded to other low-level computer vision tasks such as image denoising, image dehazing, and image enhancement. We will further verify the performance of MDCN on other image restoration tasks in future works.

REFERENCES

[24] Yifan Zuo, Qiang Wu, Yunming Fang, Ping An, Liqin Huang, and Zhifeng Chen. Multi-scale frequency reconstruction for guided depth map super-


[65] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans-


Juncheng Li received the B.Sc degree in Computer Science and Technology from Jiangxi Normal University, Nanchang, China, in 2016. He is currently pursuing the Ph.D degree with the School of Computer Science and Technology, East China Normal University, Shanghai, China. His main research interests includes image restoration, computer vision, deep learning, and medical image processing.

Faming Fang received the Ph.D. degree in Computer Science from East China Normal University, Shanghai, China, in 2013. He is currently a Professor with the School of Computer Science and Technology, East China Normal University. His main research area is image processing using the variational methods, PDEs and deep learning.

Jiaqian Li received the B.Sc. degree in Computer Science and Technology from Zhejiang Normal University, Jinhua, China, in 2018. She is pursuing the M.Phil degree with the School of Computer Science and Technology, East China Normal University, Shanghai, China. Her research interests includes low-light image enhancement and image processing.

Kangfu Mei received the B.Sc degree in Computer Science and Technology from Jiangxi Normal University, Nanchang, China, in 2019. He is currently pursuing the M.Phil degree with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China. His main research interests includes image generation, image processing, and computational photography.

Guixu Zhang received the Ph.D. degree from the Institute of Modern Physics, Chinese Academy of Sciences, Lanzhou, China, in 1998. He is currently a Professor with the School of Computer Science and Technology, East China Normal University, Shanghai, China. His research interests includes hyperspectral remote sensing, image processing, and artificial intelligences are image processing, machine learning and scientific computing.