IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 29, 2020, accepted November 15, 2020, date of publication November 25, 2020,
date of current version December 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3040542

Multi-Wavelet Residual Dense Convolutional Neural Network for Image Denoising

SHUO-FEI WANG, WEN-KAI YU, AND YA-XIN LI

Center for Quantum Technology Research, School of Physics, Beijing Institute of Technology, Beijing 100081, China
Key Laboratory of Advanced Optoelectronic Quantum Architecture and Measurement of Ministry of Education, School of Physics, Beijing Institute of Technology, Beijing 100081, China

Corresponding author: Wen-Kai Yu (yuwenkai@bit.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61801022, in part by the National Key Research and Development Program of China under Grant 2016YFE0131500, and in part by the Civil Space Project of China under Grant D040301.

ABSTRACT The neural networks with large receptive field show excellent fitting ability and have been successfully applied in image denoising, but with a difficulty to reduce the computational overhead while acquiring good denoising performance. Here we choose a representative of the above networks named multi-wavelet convolutional neural network (MWCNN) as the backbone. To obtain a better tradeoff between the denoising performance and computation speed, we propose to adopt residual dense blocks (RDBs) in each layer of the MWCNN. We call this scheme multi-wavelet residual dense convolutional neural network (MWRDCNN). Benefitting from the applied short-term residual learning strategy, it can increase the learning efficiency. Besides, since we use a hierarchical structure to build our network, the adopted RDBs in different layers are helpful for extracting more object details in different scales. Both horizontal and vertical comparison experiments have been performed to demonstrate the effectiveness of this network in image denoising. The results also show that our MWRDCNN takes much shorter time than other RDB-based networks to extract more features from adjacent layers and is good at handling the images which are badly corrupted by the noise. Thereby, it is a successful attempt to make full use of the advantages of multiple networks without any conflicts.

INDEX TERMS Image denoising, receptive field, residual dense block, convolutional neural network.

I. INTRODUCTION

Image denoising is one of well-known ill-posed problems. The natural images are typically corrupted by the external noise like electromagnet wave interruption [1] or the internal noise of detectors [2], while the synthetic aperture radar images often suffer from the speckle noise due to coherent imaging mechanisms [3]. For practical application requirements, many image denoising algorithms have been developed [3]–[9]. In recent years, thanks to the increasing computing power brought by the advanced graphic processing units (GPUs), a variety of neural networks, especially convolutional neural network (CNN), have sprung up to deal with image denoising tasks. In the CNN, the degraded image can be treated as a nonlinear mapping of the original image [10]. By using the nonlinear activation function, the CNN is gifted in handling image denoising problems. Once the network is trained, the image denoising procedure can be accomplished via a simple forward propagation, thus it can significantly reduce the computation time.

During this decade, many kinds of convolutional neural networks have been developed, most of them can be divided into three categories in terms of their structural features: simple, multi-residual and U-net structures. The simple ones do not have skip connections, and their feature mappings are performed layer-by-layer. Due to their low computational overhead and convenient design processes, the CNNs with simple structures have been widely used in early attempts [11]–[13]. However, the connections between convolution layers are neglected in these networks, and the dead neurons will be incurred because of the vanishing gradient problem in the very deep networks. In 2015, residual learning [14] was realized by adopting shortcut connections (i.e., the element-wise addition) between inputs and outputs. This kind of networks is of the multi-residual structure, which is helpful for maintaining suitable gradients in the back propagation and providing the possibility of obtaining a better performance in the deep networks. After that, several networks

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

with more complicated connections have been put forward to take full advantage of the relationship between each layer [15]–[17]. As for the U-net structure, it gets its name from its special architecture whose backbone is composed of the pooling and up-convolution layers and was first proposed by Ronneberger *et al.* [18] for biological segmentation. With the help of the pooling layers, the U-net structure can significantly reduce the calculation cost. Besides, the U-net structure establishes either element-wise addition or concatenation connections between the layers in the same level. Recently, Liu *et al.* [10] proposed to adopt the discrete wavelet transform (DWT) and inverse discrete wavelet transform (iDWT) in the U-net structure to eliminate information loss caused by the pooling operations. This multi-level wavelet convolutional neural network (MWCNN) [10] enhances the performance of the deep learning techniques in image denoising tasks. The MWCNN has a very good U-net structure and can be regarded as a typical representative of the highest level of the U-net [10].

To some extent, the multi-residual and U-net structures are the optimized variants of the simple structures. In this work, we will choose 8 representative networks as references, including (1) the simple structures: the super-resolution convolutional network (SRCNN) [19], the learning deep CNN denoiser prior for image restoration (IRCNN) [20], the denoising convolutional neural network (DnCNN) [21], the very deep super-resolution convolutional network (VDSR) [22]; (2) the multi-residual structures: the residual encoder-decoder network with 30 layers (RED30) [17], the residual dense network (RDN) with 32 feature map channels in front of each residual dense block (RDB) (6 RDBs in total) which contains 10 convolutional layers (G32C6D10) [23], the memory network (MemNet) [24]; and (3) the U-net structure: the MWCNN [10]. Generally, these reported networks were trained with different training sets and parameters [25]. To make a fair comparison, we will train these representative networks in the same condition. As shown in Fig. 1, both the multi-residual and U-net structures have higher peak signal-to-noise ratios (PSNRs) than the simple structures. Nevertheless, the multi-residual structures need to take enormous computation time, while the U-net structures have fewer internal connections and less efficient residual learning. Given this, here we propose a more robust CNN by combining both the MWCNN and RDBs together, so it can be categorized into the U-net & multi-residual structure. We name it multi-wavelet residual dense convolutional network (MWRDCNN).

The main innovation points of this work include:

- deducing the convolution and inverse convolution representations of the sub-images' decomposition (in DWT) and composition (in iDWT), respectively, to enhance the portability of the algorithm;
- adopting residual dense blocks (RDBs) in our network to increase the learning efficiency and improve the denoising performance, i.e., utilizing the short-term residual learning strategy in this network;

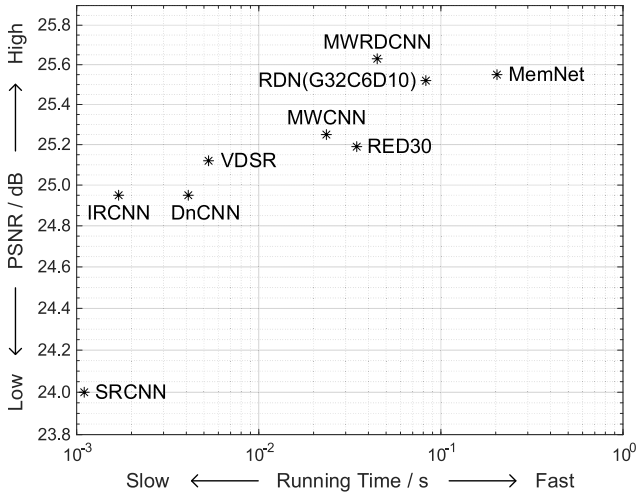

FIGURE 1. The denoising performance of our MWRDCNN and 8 state-of-the-art convolutional neural networks by using the training sets Set14, with the standard deviation σ of the additive Gaussian noise being 75. The average peak signal-to-noise ratios (PSNRs) and running time for the same single test image (with the gray values ranging from 0 to 255) were evaluated with an RTX 2080Ti GPU.

- applying RDBs in different layers to extract more object details in different scales.

The main contributions of this work include:

- validating the effectiveness of the RDBs in image denoising tasks;
- demonstrating that our MWRDCNN architecture is superior to one classic denoising algorithm (once thought to be the best) and eight state-of-the-art networks in removing the Gaussian noise;
- giving extensive performance comparisons among these networks in presence of four different types of noise.

II. RELATED WORKS

A. TRADITIONAL DENOISING ALGORITHMS AND EARLY CNN STRUCTURES FOR IMAGE DENOISING

It is worth mentioning that before the CNNs were applied in image denoising tasks, the conventional methods were relatively mature [26]–[30]. In 2007, Dabov *et al.* [8] proposed the block-matching and 3D filtering (BM3D) method, which was once thought to be the most efficient denoising algorithm. So in the early years when the CNNs were adopted to solve the ill-posed problems of image denoising, the aforesaid BM3D method was often treated as the reference criteria for the performance evaluation of newly proposed methods. In 2008, Jain and Seung [31] proposed a four-layer convolutional network, which was proven to have a more superior performance than conventional wavelet and Markov random field (MRF) methods [32]. Later, Xie *et al.* [33] developed a deep network scheme utilizing the stacked sparse denoising auto-encoders, which could achieve a comparable performance with the K-singular value decomposition (K-SVD) algorithm [34]. Although these CNN-based methods transcend most of classic algorithms, they still fail to achieve better results than the BM3D method. The monopoly of the BM3D was broken in 2014 by a three-layer fully

convolutional network (FCN) proposed by Dong *et al.* [19]. After that, Zhang *et al.* enlarged the depth of the network to 17 layers [21], with the performance being greatly increased. It motivates the CNNs to develop towards very deep architectures [35]–[37].

B. RESIDUAL LEARNING

A typical method to achieve residual learning is to adopt addition operations at the end of the network, which is expected to be used to solve the vanishing gradient problem. However, as the depth of the network grows, it becomes much harder for the CNN to keep a long-term memory. Fortunately, several intuitive solutions have been proposed to establish more complicated connections among convolutional layers. For instance, Mao *et al.* [17] considered to build multiple connections between the encoders and decoders by using element-wise addition; Tai *et al.* [24] adopted the dense connected memory blocks to take into account both the short-term memory and long-term memory; Zhang *et al.* [23] used the residual dense block to make full use of all convolutional layers.

C. U-NET ARCHITECTURE AND DWT IN THE CNNs

To avoid overfitting and to reduce the amount of calculation, the pooling layers are commonly used in the CNNs. This means that the sizes of feature maps reduce with the forward propagation. Early CNNs were used to mainly solve the classification problems by offering each category a single label. However, the image processing tasks (like image segmentation and denoising) are very different from the classification problems: the formers require the outputs to be the images of almost the same sizes as those of the inputs. In 2015, Ronneberger *et al.* [18] adopted the up-convolutional layers, which enlarged the sizes of feature maps via convolution and extended CNN's application to the biomedical segmentation field. Such novel network architecture was called U-net, in which the calculations were accelerated, and the pooling layers were used to enlarge the receptive field (RF). Therefore, the CNNs with U-net structures seem to be more suitable for dealing with image denoising tasks compared with traditional networks. Nevertheless, the pooling operation will inevitably result in loss of information. Inspired by conventional wavelet algorithms, Bae *et al.* [38] put forward a wavelet residual network (WavResNet), which adopted the DWT and iDWT layers instead of the pooling layers. Later, Liu *et al.* [10] optimized this network on the basis of the U-net and proposed the MWCNN to further improve the image denoising performance. It can be seen that the above works succeed in comprehensive learning, calculation simplifying and RF enlarging. To absorb the advantages of these networks, we propose the MWRDCNN here. The details of this network will be given in the next section.

III. PROPOSED METHOD

In this section, we will first briefly review the procedures of the DWT and iDWT to establish theoretical basis of our method. Then we will formally describe our MWRDCNN, which is based on the DWT, iDWT and RDBs. The details of RDBs will be also given. After that, the implementation details and its differences from the previous works will be presented.

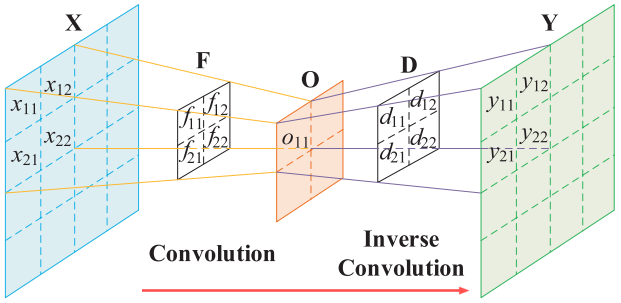

FIGURE 2. Schematic diagram of the convolution and inverse convolution processes.

A. DWT AND iDWT

Before introducing the DWT and iDWT, we will first draw the schematic diagram of the convolution and inverse convolution processes involved in the CNNs, as shown in Fig. 2. Taking the four pixels in the top left corner of the input image $\mathbf{X}$ as an example, when they are convolved with a 2×2 filter $\mathbf{F}$, we will have the first pixel value in the hidden layer $\mathbf{O}$ via

$$o_{11} = \sum_{i=1}^{2}\sum_{j=1}^{2} x_{ij} \times f_{ij}. \tag{1}$$

In the direction of forward propagation, the pixel values of the unknown output image $\mathbf{Y}$ can be obtained by

$$y_{ij} = o_{11} \times d_{ij}, \quad i, j \in [1, 2]. \tag{2}$$

By performing the same operation for the rest pixels with a stride of 2, we can get the output image $\mathbf{Y}$ via the inverse convolution between $\mathbf{O}$ and a 2×2 inverse filter $\mathbf{D}$:

$$\mathbf{Y} = iConv(\mathbf{O}, \mathbf{D}), \tag{3}$$

where *iConv* represents the inverse convolution operation. From the hidden layer's point of view, o_{11} can also be symmetrically regarded as the convolution result of the four pixels in the top left corner of the output image $\mathbf{Y}$ and the dual filter $\mathbf{D}'$ of $\mathbf{D}$, as illustrated in Fig. 2.

Now, let us review the concepts of the DWT and iDWT, which are very common tools in image processing.

A given image $\mathbf{x}$ can be decomposed into four sub-images by DWT, i.e. the low-pass image $\mathbf{x}_{\mathrm{A}}$ (average) and three high-pass images including $\mathbf{x}_{\mathrm{H}}$ (horizontal), $\mathbf{x}_{\mathrm{V}}$ (vertical), $\mathbf{x}_{\mathrm{D}}$ (diagonal). From some perspective, the process of the DWT can be seen as a convolution operation between $\mathbf{x}$ and four 2×2 filters (as described below, in a stride of 2):

$$\mathbf{f}_{\mathrm{A}} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{f}_{\mathrm{H}} = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix},$$
$$\mathbf{f}_{\mathrm{V}} = \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{f}_{\mathrm{D}} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{4}$$

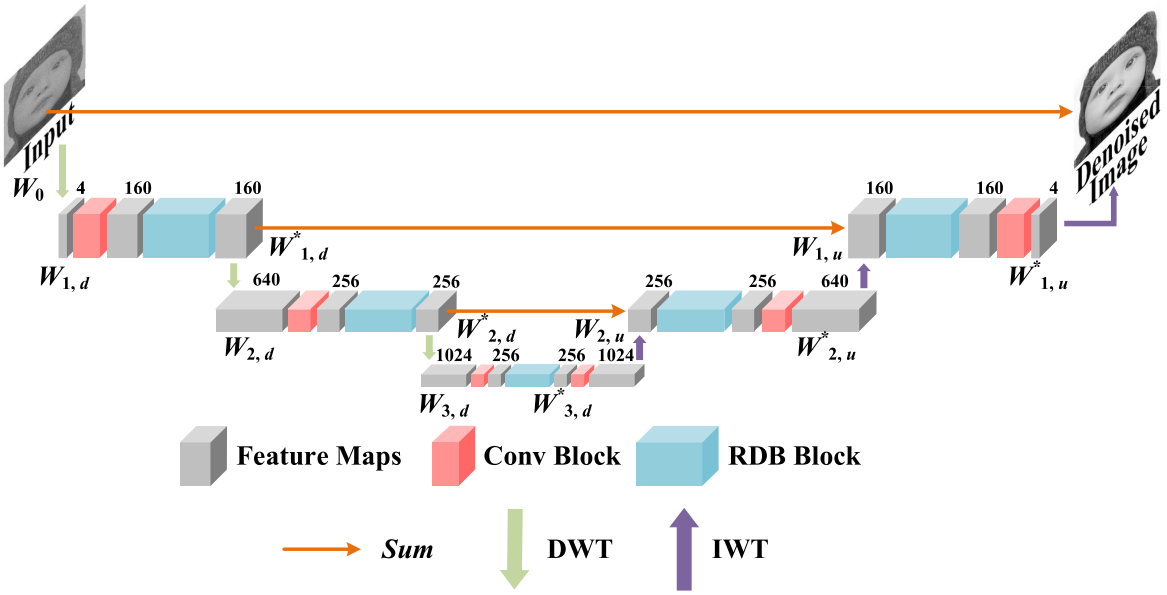

FIGURE 3. MWRDCNN architecture. The numbers of channels are annotated on the top of the multi-channel feature map boxes and the names of the representative feature maps are given under the boxes.

Thereby, to perform the DWT process, we can use the following formula to obtain four decomposed sub-images

$$\mathbf{x}_i = Conv\left(\mathbf{x}, \mathbf{f}_i\right), \quad i = \text{A, H, V or D}, \tag{5}$$

where *Conv* refers to the convolution operation, $\mathbf{x}_i$ can be treated as the feature maps, and $\mathbf{f}_i$ denote the filters. The pixel-sizes of four sub-images are all half of the input image's size. To some extent, the DWT has the similar downsampling effect as the pooling operation [18] in the U-net structure.

Due to the orthogonality of four filters, there is no information loss during the downsampling process, which means that the target image can be completely recovered by performing the iDWT. Previously, Liu *et al.* [10] adopted four equations to describe the iDWT process. Here we find the iDWT can also be expressed as an inverse convolution operation:

$$\tilde{\mathbf{x}} = Sum\left(iConv\left(\mathbf{x}_i, \mathbf{f}_i/4\right)\right), \ i = \text{A, H, V and D}, \tag{6}$$

where *Sum* denotes the element-wise addition. By this means, the sub-images' decomposition (in DWT) and composition (in iDWT) processes can be performed via the convolution and deconvolution operations, which are generally included in commonly used deep learning algorithm packages. This provides more convenience for algorithm programming.

Considering the foregoing information lossless property, the DWT and iDWT are gradually utilized in the CNNs [39]. Among them, the MWCNN is one of the most representative networks [10]. Therefore, here we adopt the MWCNN as the framework of our network. The channel number of feature maps is set to c. Assuming the sizes of the feature maps before the DWT are all $h \times w$, then the sizes of the decomposed images are all $\frac{h}{2} \times \frac{w}{2}$. Several pairs of the DWT and iDWT operations are sequentially performed to establish the hierarchical architecture, which will be detailed soon.

B. NETWORK ARCHITECTURE

As shown in Fig. 3, we retain the backbone of the MWCNN and use the RDB to optimize its each level. By this means, we build our MWRDCNN with multiple levels. For a three-level MWRDCNN, the input image can be denoted as W_0. The first DWT will decompose W_0 into four sub-band feature maps (all marked as W_1). Since these feature maps locate in the downsampling procedure, we further indicate them as $W_{1,d}$. Then the sub-band images in the upsampling procedure corresponding to $W_{1,d}$ will all be represented as $W_{1,u}$. For the first DWT, we have

$$W_{1,d} = DWT(W_0). \tag{7}$$

After that, a single convolution block (i.e., the Conv block in Fig. 3) is deployed to reduce the channel number of the feature maps for improving the inter-band independency of feature maps [39] and decreasing the computation time. Whether the block contains a batch norm layer or not depends on the number of the current layer. After the sub-band images pass through the RDB, we will obtain $W^*_{1,d}$ via

$$W^*_{1,d} = RDB(CB(W_{1,d})), \tag{8}$$

where *CB* denotes the function of the convolution block. After accomplishing the local feature fusion via the RDB, the second DWT is adopted to produce a hierarchical architecture in this network, where the sub-band images in the second level can be written as

$$W_{2,d} = DWT(W^*_{1,d}). \tag{9}$$

According to this procedure, the feature maps in the ith level during the downsampling procedure can be derived from the following equations

$$\begin{cases} W_{i,d} = DWT(W^*_{i-1,d}), \\ W^*_{i,d} = RDB(CB(W_{i,d})). \end{cases} \tag{10}$$

Suppose that the network contains n levels, the upsampling process begins after the feature maps passing the nth RDB. To guarantee the symmetric of this architecture, another convolution block needs to be used before the first iDWT operation. The iDWT in the highest level can be described as

$$W_{n-1,u} = iDWT(CB(W^*_{n,d})), \tag{11}$$

where $W_{n-1,u}$ refers to the $(n-1)$th upsampled feature maps. Then the element-wise addition is adopted to establish long-term residual learning between the feature maps $W_{n-1,u}$ and $W^*_{n-1,u}$ in the same level. In each layer of the downsampling procedure, the convolution block comes first, then followed by the RDB; while during the upsampling process, the order of the convolution block and RDB in each layer needs to be reversed, i.e., set the RDB first, then deploy the convolution block. As mentioned earlier, the role of the convolution blocks in the downsampling process is to reduce the number of channels, while the convolution blocks in the upsampling procedure have the opposite effect, i.e., to increase the number of channels to equal the channel number of corresponding feature maps in the downsampling procedure of the same level, in order to facilitate the subsequent element-wise addition operations. Subsequently, we will obtain the output feature maps $W^*_{n-1,u}$ via

$$W^*_{n-1,u} = CB(RDB(Sum(W_{n-1,u}, W^*_{n-1,u}))). \tag{12}$$

Similar to Eq. (10), the formulas in the upsampling procedure can be expressed as

$$\begin{cases} W_{i-1,u} = iDWT(W^*_{i,u}), \\ W^*_{i-1,u} = CB(RDB(Sum(W_{i-1,u}, W^*_{i-1,d}))). \end{cases} \tag{13}$$

It is worth noting that there are no more learnable parameters after the last convolution layer, thus adding one more rectified linear unit (ReLU) excitation layer will definitely degrade the quality of the final output image. Therefore, we set the last convolution block to be a single convolution layer.

In this work, we use the PSNR as a unitless performance measure for evaluating the quality of the final output images of the networks:

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right). \tag{14}$$

From the above definition we can see that the PSNR value depends on the mean square error (MSE), which is defined as MSE $= \frac{1}{pq}\sum_{i,j=1}^{p,q}[U_o(i,j) - \tilde{U}(i,j)]^2$, where p and q are the pixel dimensions of both the input image and output image. The MSE describes the squared distance between the denoised image $\tilde{U}(i,j)$ and the original image $U_o(i,j)$. Naturally, the larger is the PSNR value, the better is the image quality of the output result.

The quadratic term will come out a 2 during the derivative of backpropagation when training the network. To cancel out this 2, we define the loss function of our network as:

$$L_2(P) = \frac{1}{2N}\sum_{i=1}^{N} \|\Psi(\mathbf{b}_i, P) - \mathbf{x}_i\|_2^2, \tag{15}$$

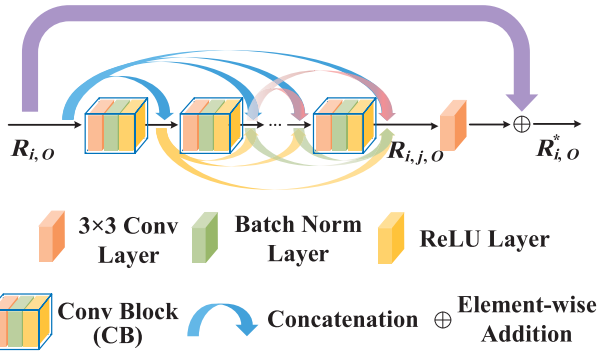

FIGURE 4. Structure of the residual dense blocks in our MWRDCNN.

where N is the batch size; $\Psi(\mathbf{b}_i, P)$ refers to the output image of the network; P denotes the learnable parameters of the network; $\mathbf{b}_i$ and $\mathbf{x}_i$ are the image corrupted by the additive noise (treated as the input image of the network) and the original image (regarded as the target, also called the ground truth), respectively. A lot of pairs of the noisy images and original images form a training set $\{(\mathbf{b}_i, \mathbf{x}_i)\}_{i=1}^N$.

C. RESIDUAL DENSE BLOCK

The RDBs are generally applied to build tight connections among layers and have been widely used to address the single image super resolution (SISR) tasks [23]. In this work, we extend the application of the RDBs to the field of image denoising. The detailed architecture of the RDBs is illustrated in Fig. 4. The inputs and outputs of the RDB in the ith level are denoted as $R_{i,o}$ and $R^*_{i,o}$, respectively, where $o = d$ (downsampling) or u (upsampling). Then we can write their relationship as below:

$$R^*_{i,o} = RDB(R_{i,o}). \tag{16}$$

For a RDB with j convolution blocks, the output of the final concatenation result $R_{i,j,o}$ can be obtained via

$$R_{i,j,o} = Concat(CB(R_{i,j-1,o}), R_{i,j-1,o}, \cdots, R_{i,2,o}, R_{i,o}), \tag{17}$$

where *Concat* denotes the concatenation operation. Assuming the channel number of $R_{i,o}$ is c, then the $R_{i,j,o}$ will have $j \times c$ channels. Obviously, $R_{i,j,o}$ has more channels than $R_{i,o}$. Thereby, another convolution layer is needed to added onto $R_{i,j,o}$ to reduce its channel number to equal that of $R_{i,o}$, in order to facilitate the subsequent element-wise addition operations and make the residual learning in this RDB go smoothly. Then we can calculate $R^*_{i,o}$ through

$$R^*_{i,o} = Sum(R_{i,o}, Conv(R_{i,j,o})). \tag{18}$$

D. IMPLEMENTATION DETAILS

Without loss of generality, we build three hierarchical levels to describe the implementation details of our MWRDCNN. Each RDB contains three convolution blocks. The sizes of the convolution kernels in the convolution layers are all set to 3×3, and their weights need to be adjusted in the backpropagation; while the sizes of the convolution kernels in the DWT and iDWT are all 2×2, but their weights will not be adjusted

in the backpropagation. It is worth mentioning that there are no convolution layers in the DWT and iDWT operations. The numbers of channels in each layer are marked in Fig. 3.

E. DISCUSSION

Next, we will show the benefits of this proposed MWRDCNN and explain how we address the trade-off relationship between the performance and time consumption by combining the preeminent structures of both MWCNN and RDN.

1) DIFFERENCES BETWEEN THE RDN AND OUR MWRDCNN

The RDN does not contain the U-net structure, so there are only two convolution layers in the RDN. In our MWRDCNN, the sub-band images are new feature maps decomposed from the input images. To extract the features, we need to add one more convolution block in each level of this network.

2) DIFFERENCES BETWEEN THE MWCNN AND OUR MWRDCNN

Here we summarize three main differences between the MWCNN and our network. In the MWCNN, only long-term residual learning is applied by element-wise adding the feature maps in both downsampling and upsampling procedures of the same level. Since we adopt the RDBs in each layer of our MWRDCNN, both the short-term and long-term connections are established, resulting in a more efficient learning mechanism. And the RDBs adopted in different layers of the hierarchical structure help extract more object details from different scales. Additionally, whether in the downsampling or upsampling procedure, the highest level of our network only has a half of the number of convolutional blocks compared with the MWCNN. By this means, the MWRDCNN can accelerate the computation process.

IV. EXPERIMENTS

In this section, we will first introduce the noise types and training settings, and then provide the detailed descriptions of the training sets and training procedure. Considering that the networks to be compared were generally trained with different patch sizes, we will conduct an experiment to discuss whether the performance is influenced by the patch size. After that, some fair comparisons will be made to demonstrate the overall performance of our network, by using the running time and the quantitative evaluators (PSNR and structural similarity index measure (SSIM) [40]). The qualities of the denoised images are also given. In this work, the images used in both the training sets and test sets are of grayscale.

A. EXPERIMENTAL SETTINGS

1) TRAINING AND TESTING INSTRUCTIONS

The DIV2K (one of the most popular training sets [10], [15], [25] benefiting from its high quality) is used here. It contains 800 images for training, 100 images for validation, and 100 images for testing.

In our experiments, four classic types of the additive noise are taken into account, including the Gaussian, Rayleigh, uniformly and exponentially distributed noise. That is, a noisy image is the sum of an original image and the noise. Here the gray values of the original image are real numbers ranging from 0 to 1. The probability density functions of these four types of noise can be written as

$$noise_{\text{Gaussian}} = \frac{255}{\sigma\sqrt{2\pi}} e^{-\frac{(255x)^2}{2\sigma^2}}, \text{ where } x \in (-\infty, \infty); \quad (19)$$

$$noise_{\text{Rayleigh}} = \frac{x}{\lambda^2} e^{-\frac{x^2}{2\lambda^2}}, \text{ where } x \in (0, \infty); \quad (20)$$

$$noise_{\text{Uniform}} = \frac{1}{a}, \text{ where } x \in (0, a); \quad (21)$$

$$noise_{\text{Exponential}} = \frac{1}{\theta} e^{-\frac{x}{\theta}}, \text{ where } \in (0, \infty). \quad (22)$$

Here, σ denotes the standard deviation of the Gaussian noise, a is the right margin of the uniformly distributed noise, and θ is the rate parameter of the exponentially distributed noise. If two independent vectors obey the normal distribution with the same standard deviation λ, the norm of these two vectors will obey the Rayleigh distribution in Eq. (20). The variable x in these equations refers to the gray scale of the noise.

In this work, we choose one traditional method (BM3D) and eight other different CNNs (SRCNN, IRCNN, DnCNN, VDSR, RED30, MWCNN, RDN and MemNet) as our main competitors, and accomplish both the training and testing procedures on a NVIDIA RTX 2080Ti GPU with 11 GB memory. To evaluate the image denoising effect, we perform two sets of experiments: the horizontal and vertical comparison experiments. The patch size (the size of each image in the training set) and batch sizes (the number of the training images in one iteration) selected in the first set of experiments will make our existing hardware's computing resources operate at full capacity to horizontally examine the maximum denoising ability of each network. The horizontal performance comparisons of these networks will be made under different standard deviations of the Gaussian noise. The second set of experiments is to vertically compare the image denoising capabilities of different networks in presence of the noise that follows different distributions. Due to the large number of heavy training tasks, in this set of experiments we will select completely consistent patch and batch sizes that take up a smaller amount of computing resources than the first set of experiments. Therefore, the second set of experiments can be regarded as a supplement of the first one, increasing the dimension of comparisons. Through these two sets of experiments, we will demonstrate the image denoising performance of our network in an all-round way.

In the first set of experiments, we adopt long-term training with well-designed patch and batch sizes, as shown in Table 1. We first set the batch size of our MWRDCNN to 32, which is an appropriate and commonly used parameter. To make full use of the GPU's memory size, its patch size was set to 152×152. We accordingly cropped 40000 patches form DIV2K as the training set and ensure all features can be learned. Since the SRCNN, IRCNN, DnCNN, VDSR, and MWCNN all take less memory, we adopted the same batch size 32, the same patch size 152×152 and the same training set for them. Due to the larger memory cost of the RED30,

TABLE 1. Patch and batch sizes for training different networks.

Methods	Patch size	Batch size
SRCNN [19]	152 × 152	32
IRCNN [20]	152 × 152	32
DnCNN [21]	152 × 152	32
VDSR [22]	152 × 152	32
RED30 [17]	152 × 152	21
MWCNN [10]	152 × 152	32
RDN [23]	76 × 76	28
MemNet [24]	76 × 76	18
MWRDCNN	152 × 152	32

we set its batch size to 21 (which is still in a reasonable range) and set its patch size to the same 152 × 152. The structures of the RDN and MemNet are much more complex. When the RDN and MemNet use a patch size of 152 × 152, their batch sizes can be only set to 4 or 5 (near the upper limits of our hardware), which are very small. Considering the overall training efficiency of both the RDN and MemNet, we crop every 152 × 152 patch into four 76 × 76 patches to adjust their batch sizes to more suitable values, i.e., 28 and 18. In this way, the patch and batch sizes for the first set of experiments are designed to be as close to each other as possible under the condition of ensuring almost the same training efficiency.

In the second set of experiments, in view of the large number of training tasks, if we still use the same training conditions as the first set of experiments, it will undoubtedly increase the training time overhead of the vertical comparisons. To shorten the training time and without loss of generality, we keep 40000 cropped patches unchanged, but each of size 72 × 72, and use them as the training set. The batch sizes are all adjusted to 20. The epoch number is also set to 20 to reduce the training time.

The performance of the networks are assessed by using five commonly used test sets: Set5 [41], Set12 [21], Set14 [42], BSD68 [43] and Urban100 [44].

2) OTHER PARAMETERS

To compare the proposed method with previous networks as fair as possible, we adopt the same training parameters. The adaptive moment (Adam) estimation algorithm is chosen as the optimizer, with the settings $\alpha = 0.01$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, which are used in many other training processes [10], [17], [23].

The learning rates for the proposed method are carefully adjusted to guarantee that the losses are appropriately reduced. For the first set of experiments, there are 3 stages (45 epochs in total). In the first 15 epochs, the learning rates are fixed to 10^{-3}. In the next 20 epochs, the learning rates decay exponentially from $10^{-3.8}$ to 10^{-4}. In the last 10 epochs, the learning rates decay exponentially from $10^{-4.5}$ to 10^{-5}. While in the second set of experiments, the learning rates decay exponentially from $10^{-3.5}$ to 10^{-5}.

All networks are trained with the same parameters described above to ensure that the testing performance only depends on the properties of the networks. The rotation and flip of the patches are used here for data augmentation. The training and testing procedures are accomplished with a package named MatConvNet [45].

TABLE 2. Average PSNR (dB) and SSIM values of the SRCNN with different patch sizes.

Test set	patch size = 76 × 76	patch size = 152 × 152
Set5	27.20 / 0.7569	27.15 / 0.7544
Set12	26.11 / 0.7336	26.06 / 0.7300
Set14	25.84 / 0.6873	25.80 / 0.6856
BSD68	25.50 / 0.6706	25.48 / 0.6691
Urban100	24.68 / 0.7208	24.60 / 0.7177

B. DISCUSSION ON THE DENOISING PERFORMANCE OF USING DIFFERENT PATCH SIZES

Since we use two training sets with different patch sizes, i.e., 76 × 76 and 152 × 152, in this section we will conduct a small experiment to discuss the effect of the patch size on the denoising performance. Without loss of generality, here we will directly test on the SRCNN because it takes the least training time. It is trained by using two training sets mentioned in Subsection IV.A and the same parameters as the first set of experiments. The standard deviation σ of the Guassian noise is set to 50, and the denoising performance of the SRCNN is presented in Table 2.

It can be seen that the PSNRs and SSIMs of the SRCNN corresponding to different patch sizes are almost the same (their differences can be neglected), from which we can infer that the change of patch size will not affect the final denoising effect.

C. RUNNING TIME OF THE TEST PROCEDURE

The computational efficiency of the network plays an important role in the performance evaluation. In this work, a CuDNN-v7.5 deep learning library with CUDA 10.1 is adopted to build the test environment under 64-bit Windows 10 operating system. The time cost of the denoising procedure mainly depends on the complexity of the network, the amount and size of the test sets, which means that the specific training conditions and specific noise types (distributions) have little impact on the running time of the test procedure. Therefore, we select the data corresponding to the Gaussian noise with a standard deviation of 50 in the first set of experiments as a basis reference. After finishing the computational processes with the NVIDIA RTX 2080ti GPU, we record the average testing time of different networks under the same test environment, as shown in Table 3.

Benefiting from the DWT and iDWT (which can reduce the sizes of the feature maps), our MWRDCNN runs significantly faster than other two RDB-structured networks, i.e., the RDN and MemNet. Although the running time of the MWRDCNN is longer than those of the SRCNN, IRCNN, DnCNN, VDSR, RED30 and MWCNN, the high-quality reconstruction of the MWRDCNN (which will be demonstrated in the following two sets of experiments) is sufficient to make up for the disadvantage of time consumption.

TABLE 3. Running time (in seconds) of different networks in image denoising tasks. The standard deviation of the additive Gaussian noise is set to $\sigma = 50$.

Methods	Set5	Set12	Set14	BSD68	Urban100
SRCNN [19]	0.0011	0.0011	0.0011	0.0011	0.0211
IRCNN [20]	0.0017	0.0017	0.0018	0.0017	0.0276
DnCNN [21]	0.0041	0.0041	0.0042	0.0041	0.0430
VDSR [22]	0.0056	0.0053	0.0055	0.0056	0.0488
RED30 [17]	0.0360	0.0345	0.0652	0.0364	0.2146
MWCNN [10]	0.0240	0.0235	0.0394	0.0227	0.1248
RDN [23]	0.0875	0.0827	0.1746	0.0868	0.8788
MemNet [24]	0.2127	0.2032	0.3886	0.2409	1.4648
MWRDCNN	0.0649	0.0447	0.0993	0.0478	0.3191

D. HORIZONTAL COMPARISONS OF THE GAUSSIAN DENOISING EXPERIMENTS

The results of the proposed MWRDCNN and 9 previous methods under the additive Gaussian noise of different standard deviations are presented in Table 4. The highest score in each row is highlighted in bold. It can be seen that when the standard deviation σ of the additive Gaussian noise is low, the performance of our MWRDCNN is only slightly inferior to the MemNet. As the standard deviation σ increases, the denoising performance of our MWRDCNN gradually exceeds that of the MemNet.

It is worth noting that our MWRDCNN surpasses the MWCNN (the basis of our network) $\sim$ 0.1 dB in the first four test sets and $\sim$ 0.3 dB in the test set Urban100. It can be concluded that the proposed network does benefit from the residual learning brought by the adopted RDBs.

Particularly, the networks with RDBs (i.e., the RDN, MemNet and MWRDCNN) all perform much better than the others in the test set Urban100. Notice that many images in this set contain various grids, the RDBs are gifted to deal with this kind of complicated patterns. On the other hand, when the standard deviation σ is large, our MWRDCNN performs better with the vast majority of test sets due to the use of the DWT and iDWT operations. It indicates that the DWT and iDWT operations are helpful for improving the image denoising performance of the network, by enlarging the RF of the network (i.e., improving the fitting ability) and avoiding the information loss of feature maps.

In conclusion, our MWRDCNN combines the advantages of both the MWCNN and RDB together, and is competent in image denoising.

E. VERTICAL COMPARISONS OF DIFFERENT NOISE EXPERIMENTS

Next, we will both quantitatively and qualitatively evaluate the performance of each network under different types of noise. The chosen networks include the IRCNN, DnCNN, RED30, MWCNN, RDN, MemNet and our MWRDCNN. For each figure, we will enlarge its two cropped patches for comparison. The PSNR values of the whole images obtained by different networks are listed below the corresponding patches. To reduce the total testing time, we only adopt the test set Set12 in this set of experiments.

1) GAUSSIAN NOISE

Table 5 shows the denoising results with respect to the Gaussian noise. It can be seen that our MWRDCNN still exhibits the highest scores after a shorter-time training (compared with the first set of experiments). The score gap between our method and other 6 methods widens as the standard deviation of the Gaussian noise increases.

The qualitative comparison results of the ''10'' image are given in Fig. 5. We choose the mast in the red rectangle and the wave in the green rectangle to compare the denoising performance for tiny details. It can be seen that our MWRDCNN can correctly recover the profile in the red rectangle, especially the right mast. Our major competitor RDN and MemNet reconstruct the image too smooth that the masts can be hardly recognized. The images reconstructed by other methods also show blurred profiles with missing

TABLE 4. Average PSNR (dB) and SSIM values of one traditional method, eight state-of-the-art methods and our MWRDCNN in the Guassian denoising experiments. The structures of the networks are classified in this table. The proposed network has both the multi-residual and the U-net structures.

Test set	σ	Traditional Method	Simple Structures				Multi-Residual Structures			U-Net Structures	Proposed
		BM3D [8]	SRCNN [19]	IRCNN [20]	DnCNN [21]	VDSR [22]	RED30 [17]	RDN [23]	MemNet [24]	MWCNN [10]	MWRDCNN
Set5	25	30.67 / 0.8604	30.36 / 0.8433	31.17 / 0.8696	31.23 / 0.8709	31.32 / 0.8731	31.43 / 0.8655	31.47 / 0.8773	31.51 / 0.8779	31.47 / 0.8680	**31.55/0.8694**
	50	27.52 / 0.7866	27.15 / 0.7544	28.08 / 0.8009	28.13 / 0.8026	28.23 / 0.8062	28.51 / 0.7999	28.46 / 0.8152	28.49 / 0.8172	28.63 / 0.8055	**28.70/0.8079**
	75	25.70 / 0.7272	25.23 / 0.6889	26.21 / 0.7485	26.24 / 0.7504	26.38 / 0.7550	26.42 / 0.7598	26.69 / 0.7711	26.72 / 0.7729	26.53 / 0.7660	**26.79/0.7759**
Set12	25	29.97 / 0.8505	29.46 / 0.8315	30.31 / 0.8589	30.38 / 0.8602	30.48 / 0.8633	30.60 / 0.8663	30.67 / 0.8681	**30.74/0.8696**	30.61 / 0.8678	30.71 / 0.8696
	50	26.72 / 0.7676	26.06 / 0.7300	27.09 / 0.7799	27.14 / 0.7816	27.26 / 0.7860	27.44 / 0.7938	27.55 / 0.7977	27.62 / 0.7999	27.56 / 0.7990	**27.66/0.8023**
	75	24.90 / 0.7059	24.14 / 0.6545	25.18 / 0.7189	25.18 / 0.7196	25.36 / 0.7262	25.45 / 0.7315	25.80 / 0.7471	25.82 / 0.7484	25.45 / 0.7369	**25.90/0.7526**
Set14	25	29.56 / 0.8225	29.09 / 0.8051	29.91 / 0.8354	29.96 / 0.8363	30.08 / 0.8394	30.20 / 0.8422	30.26 / 0.8439	30.30 / 0.8450	30.22 / 0.8441	**30.30/0.8463**
	50	26.37 / 0.7186	25.80 / 0.6856	26.78 / 0.7359	26.80 / 0.7366	26.92 / 0.7423	27.11 / 0.7514	27.21 / 0.7547	27.26 / 0.7557	27.21 / 0.7579	**27.32/0.7613**
	75	24.66 / 0.6503	24.00 / 0.6059	24.95 / 0.6647	24.95 / 0.6650	25.12 / 0.6727	25.19 / 0.6770	25.52 / 0.6935	25.55 / 0.6944	25.25 / 0.6878	**25.63/0.7013**
BSD68	25	28.57 / 0.8017	28.52 / 0.7953	29.12 / 0.8244	29.16 / 0.8255	29.23 / 0.8283	29.30 / 0.8313	29.35 / 0.8325	29.37 / 0.8343	29.33 / 0.8331	**29.38/0.8351**
	50	25.62 / 0.6869	25.48 / 0.6691	26.16 / 0.7156	26.18 / 0.7171	26.25 / 0.7202	26.36 / 0.7263	26.43 / 0.7320	26.45 / 0.7327	26.44 / 0.7327	**26.50/0.7355**
	75	24.19 / 0.6216	23.83 / 0.5880	24.60 / 0.6450	24.60 / 0.6464	24.70 / 0.6525	24.74 / 0.6558	24.92 / 0.6664	24.94 / 0.6673	24.81 / 0.6619	**25.00/0.6720**
Urban100	25	29.70 / 0.8777	28.34 / 0.8431	29.71 / 0.8814	29.88 / 0.8835	30.14 / 0.8890	30.45 / 0.8953	30.62 / 0.8983	**30.78/0.9017**	30.38 / 0.8973	30.62 / 0.9010
	50	25.94 / 0.7791	24.60 /0.7177	26.09 / 0.7885	26.18 / 0.7900	26.45 / 0.7995	26.92 / 0.8157	27.21 / 0.8252	**27.40/0.8311**	27.04 / 0.8252	27.36 / 0.8333
	75	23.93 / 0.7025	22.50 / 0.6203	23.94 / 0.7092	23.96 / 0.7085	24.29 / 0.7237	24.41 / 0.7295	25.24 / 0.7661	25.35 / 0.7688	24.41 / 0.7398	**25.40/0.7757**

TABLE 5. Average PSNR (dB) and SSIM values of 6 state-of-the-art networks and our MWRDCNN under the Gaussian noise in the vertical comparison experiments.

σ	IRCNN [20]	DnCNN [21]	RED30 [17]	MWCNN [10]	RDN [23]	MemNet [24]	MWRDCNN
30	29.17/0.8313	29.07 / 0.8283	29.28 / 0.8356	29.25 / 0.8400	29.63 / 0.8464	29.69 / 0.8479	**29.69/0.8496**
40	27.80/0.8110	27.70 / 0.7923	28.05 / 0.8119	28.03 / 0.8114	28.33 / 0.8160	28.37 / 0.8184	**28.42/0.8213**
50	26.73/0.7664	26.67 / 0.7627	27.03 / 0.7783	27.11 / 0.7858	27.31 / 0.7897	27.37 / 0.7914	**27.43/0.7954**

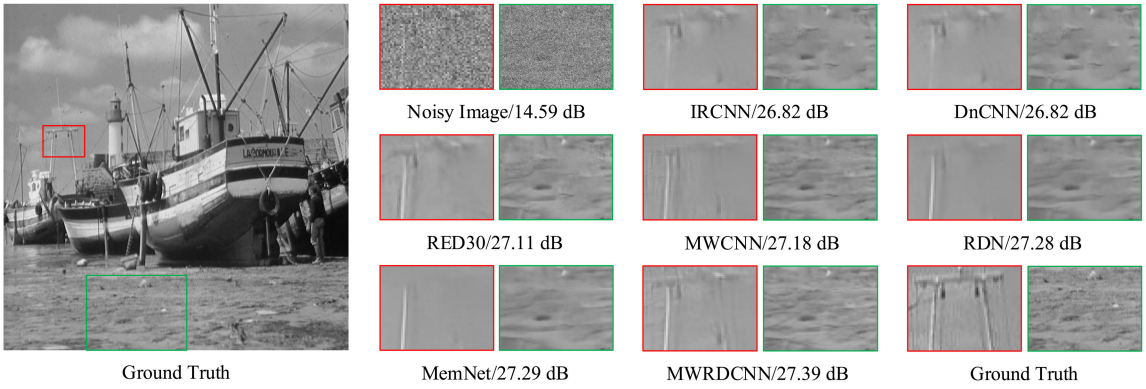

FIGURE 5. Qualitative comparisons for the image "10" in Set12 under the Gaussian noise with $\sigma = 50$.

TABLE 6. Average PSNR (dB) and SSIM values of 6 state-of-the-art networks and our MWRDCNN under the Rayleigh noise in the vertical comparison experiments.

λ	IRCNN [20]	DnCNN [21]	RED30 [17]	MWCNN [10]	RDN [23]	MemNet [24]	MWRDCNN
0.2	29.01 / 0.8321	28.84 / 0.8290	29.06 / 0.8384	28.91 / 0.8351	29.64 / 0.8509	**29.69/0.8524**	29.67 / 0.8515
0.3	27.21 / 0.7841	27.11 / 0.7828	27.60 / 0.8010	27.50 / 0.8017	27.91 / 0.8115	27.96 / 0.8130	**27.98/0.8147**
0.5	24.93 / 0.7093	24.86 / 0.7121	25.44 / 0.7378	25.54 / 0.7458	25.73 / 0.7496	25.80 / 0.7514	**25.82/0.7544**

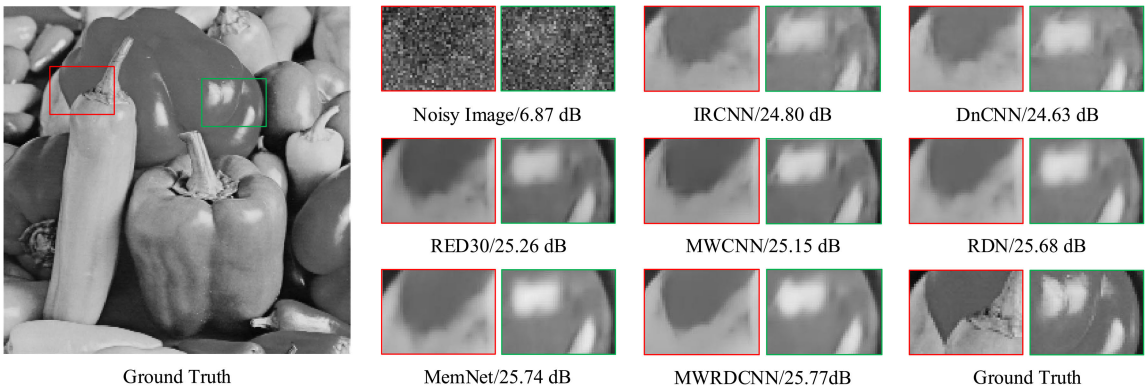

FIGURE 6. Qualitative comparisons for the image "03" in Set12 under the Rayleigh noise with $\lambda = 0.5$.

key characteristics. The same phenomenon also occurs in the green rectangles. The image recovered by MWRDCNN contains the waviest details comparing with the others. According to the quantitative and qualitative results, we find that although the PSNRs and SSIMs of the RDN and MemNet are closed to the proposed method, their reconstructed images are too smooth comparing with our MWRDCNN. It means that the DWT and the iDWT in the network do help the noisy image retain complex and subtle textures.

2) RAYLEIGH NOISE

Table 6 lists the performance of our MWRDCNN and other 6 methods in dealing with the Rayleigh noise. Compared with the predecessor MWCNN, our method increases the PSNR value by 0.3∼0.7 dB. The performance of our MWRDCNN also surpasses the MemNet in harsh noise conditions.

The qualitative comparison results (see Fig. 6) with respect to the original image "03" are also given. We choose the features with their gray values changing significantly (as shown in the red and green rectangles) to evaluate the performance of the networks in dealing with the borders of different gray levels. One can see in the red rectangles that our MWRDCNN can recover the clearest edge features. The advantage of our method in dealing with the bold lines has also been demonstrated by the enlarged results in the green rectangles.

TABLE 7. Average PSNR (dB) and SSIM values of 6 state-of-the-art networks and our MWRDCNN under the uniformly distributed noise in the vertical comparison experiments.

a	IRCNN [20]	DnCNN [21]	RED30 [17]	MWCNN [10]	RDN [23]	MemNet [24]	MWRDCNN
0.5	29.20 / 0.8439	29.20 / 0.8455	29.58 / 0.8570	29.22 / 0.8494	30.17 / 0.8698	**30.27/0.8719**	30.10 / 0.8695
1	26.41 / 0.7706	26.69 / 0.7854	27.27 / 0.8049	27.15 / 0.8014	27.77 / 0.8195	**27.83/0.8207**	27.78 / 0.8201
1.5	25.17 / 0.7371	25.24 / 0.7412	26.00 / 0.7700	25.71 / 0.7617	26.27 / 0.7795	**26.49/0.7863**	26.38 / 0.7843

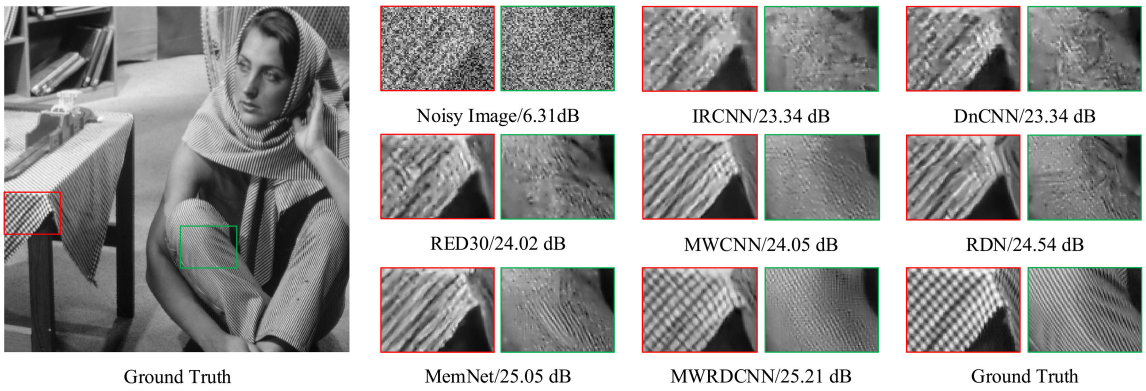

FIGURE 7. Qualitative comparisons for the image "09" in Set12 under the uniformly distributed noise with $a = 1.5$.

TABLE 8. Average PSNR (dB) and SSIM values of 6 state-of-the-art networks and our MWRDCNN under the exponentially distributed noise in the vertical comparison experiments.

θ	IRCNN [20]	DnCNN [21]	RED30 [17]	MWCNN [10]	RDN [23]	MemNet [24]	MWRDCNN
0.3	26.01 / 0.7580	25.95 / 0.7583	28.62 / 0.8392	28.25 / 0.8320	28.94 / 0.8471	**28.99/0.8491**	28.88 / 0.8481
0.4	26.84 / 0.7850	26.74 / 0.7835	27.60 / 0.8143	27.52 / 0.8149	28.01 / 0.8269	**28.07/0.8267**	27.98 / 0.8272
0.5	27.88 / 0.8159	27.78 / 0.8115	26.89 / 0.7967	26.91 / 0.8004	27.25 / 0.8082	**27.38/0.8086**	27.31 / 0.8107

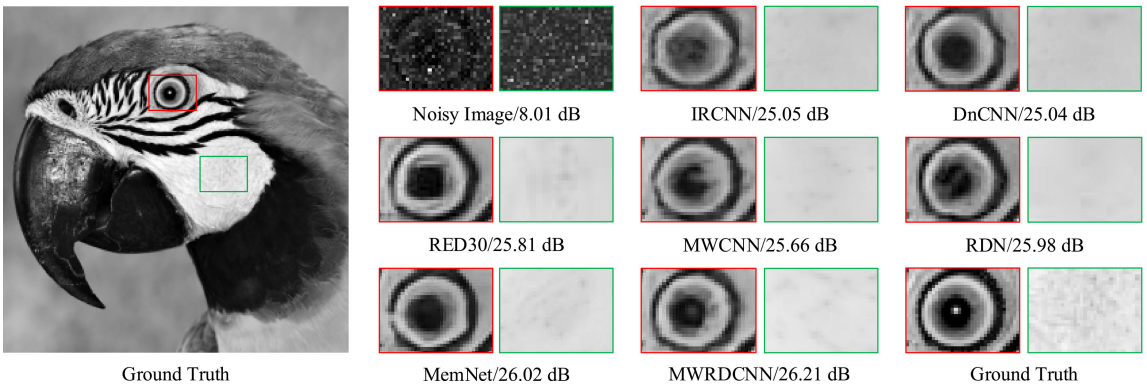

FIGURE 8. Qualitative comparisons for the image "07" in Set12 under the exponentially distributed noise with $\theta = 0.5$.

3) UNIFORMLY DISTRIBUTED NOISE

The quantitative results under the uniformly distributed noise are presented in Table 7. Compared with the networks such as the IRCNN, DnCNN, RED30 and MWCNN, our MWRDCNN outperforms them by about 0.6 $\sim$ 1.2 dB. The PSNR values of our method are only smaller than the RDN and MemNet by $\sim$ 0.1 dB. This deficiency can be compensated for with the advantage of less time consumption of our network as mentioned before.

Again, the superiority of our MWRDCNN in dealing with complex textures has been well illustrated in Fig. 7, using the original image "09". For this type of noise, we focus on the denoising performance for the textures. The chosen red and green rectangles contain abundant stripes with two or one directions, respectively. The first three methods cannot correctly restore the tablecloth textures in the red rectangles, while the MWCNN, RDN and MemNet can only roughly recover the details in one texture direction. It is clear that the results of our MWRDCNN have the highest similarity with the ground truth (i.e., the original image without noise). Noting that the stripes of the ground truth in the green rectangles are much finer than those in the red rectangles, so it is surprising to find that our MWRDCNN can also reconstruct such kind of complex and subtle textures very well.

4) EXPONENTIALLY DISTRIBUTED NOISE

Finally, we compare the performance of these networks in the case of the exponentially distributed noise, as presented in Table 8. The results are similar to those in the presence of the uniformly distributed noise.

The original image "07" in Set12 is selected here, as shown in Fig. 8. To compare the performance of different networks in dealing with the characteristics of the biological image, we mark the eye of the parrot with the red rectangle and its feathers with the green rectangle. It is worth noting that the bright dot in the pupil can be clearly seen in the recovered image of our network. However, our major competitor RDN and MemNet fail to reconstruct this tiny characteristic and the other networks can only retrieve some relatively fuzzy images. It can be clearly seen that the recovered image of our method in the green rectangles contains very subtle information. Therefore, the results in the green rectangles have also demonstrated the capability of our MWRDCNN in handling complex textures.

Overall, the MWRDCNN is promising in finely recovering the images which are badly corrupted by the noise and presents a better image denoising performance. Additionally, our MWRDCNN achieves a good balance between the computational efficiency and image denoising performance.

V. CONCLUSION

In this work, we choose the MWCNN as the backbone, and adopt the RDBs in its each downsampling and upsampling procedure to build our MWRDCNN. We prove that the DWT and iDWT in the network can be replaced by the convolution and deconvolution operations. The short-term residual learning based on RDBs strongly increases the learning efficiency. And the adopted RDBs in different layers are helpful for extracting more object details in different scales. Thus this proposed network inherits the large RF and lossless information advantages of the MWCNN brought by the use of the DWT and iDWT. The experimental results have shown that by combining these advantages, the proposed network achieves higher quality in the image denoising tasks under different types of noise. Moreover, our MWRDCNN takes shorter computation time compared with the RDB-structured networks like the RDN and MemNet. Therefore, this MWRDCNN is a successful attempt to make full use of the superiority of multiple networks. We hope that our network can be further expanded to more image processing fields in the near future, such as the SISR and image artifacts removal.

ACKNOWLEDGMENT

(Shuo-Fei Wang and Wen-Kai Yu contributed equally to this work.)

REFERENCES

[1] N. I. Matter, B. Chronik, J. M. Pauly, A. Macovski, S. M. Conolly, and G. C. Scott, "Noise performance of a precision pulsed electromagnet power supply for magnetic resonance imaging," *IEEE Trans. Med. Imag.*, vol. 27, no. 1, pp. 75–86, Jan. 2008.

[2] D. Y. Kim and K. K. O, "Reduction of NEP variations for terahertz detectors using Schottky barrier diodes in CMOS," *Electron. Lett.*, vol. 53, no. 11, pp. 732–734, May 2017.

[3] S. Liu, L. Gao, Y. Lei, M. Wang, Q. Hu, X. Ma, and Y.-D. Zhang, "SAR speckle removal using hybrid frequency modulations," *IEEE Trans. Geosci. Remote Sens.*, early access, Aug. 18, 2020, doi: 10.1109/TGRS.2020.3014130.

[4] G. Gilboa, N. Sochen, and Y. Y. Zeevi, "Image enhancement and denoising by complex diffusion processes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1020–1036, Aug. 2004.

[5] P. Coupe, P. Yger, S. Prima, P. Hellier, C. Kervrann, and C. Barillot, "An optimized blockwise nonlocal means denoising filter for 3-D magnetic resonance images," *IEEE Trans. Med. Imag.*, vol. 27, no. 4, pp. 425–441, Apr. 2008.

[6] F. Luisier and T. Blu, "SURE-LET multichannel image denoising: Interscale orthonormal wavelet thresholding," *IEEE Trans. Image Process.*, vol. 17, no. 4, pp. 482–492, Apr. 2008.

[7] Y. Le Montagner, E. D. Angelini, and J.-C. Olivo-Marin, "An unbiased risk estimator for image denoising in the presence of mixed Poisson–Gaussian noise," *IEEE Trans. Image Process.*, vol. 23, no. 3, pp. 1255–1268, Mar. 2014.

[8] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[9] S. Liu, T. Liu, L. Gao, H. Li, Q. Hu, J. Zhao, and C. Wang, "Convolutional neural network and guided filtering for SAR image denoising," *Remote Sens.*, vol. 11, no. 6, pp. 702–720, Mar. 2019.

[10] P. Liu, H. Zhang, W. Lian, and W. Zuo, "Multi-level wavelet convolutional neural networks," *IEEE Access*, vol. 7, pp. 74973–74985, Jun. 2019.

[11] P. L. Venetianer, F. Werblin, T. Roska, and L. O. Chua, "Analogic CNN algorithms for some image compression and restoration tasks," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 42, no. 5, pp. 278–284, May 1995.

[12] N. Zeng, Z. Wang, B. Zineddin, Y. Li, M. Du, L. Xiao, X. Liu, and T. Young, "Image-based quantitative analysis of gold immunochromatographic strip via cellular neural network approach," *IEEE Trans. Med. Imag.*, vol. 33, no. 5, pp. 1129–1136, May 2014.

[13] B. Zineddin, Z. Wang, and X. Liu, "Cellular neural networks, the Navier–Stokes equation, and microarray image reconstruction," *IEEE Trans. Image Process.*, vol. 20, no. 11, pp. 3296–3301, Nov. 2011.

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

[15] C. Liu, X. Sun, C. Chen, P. L. Rosin, Y. Yan, L. Jin, and X. Peng, "Multi-scale residual hierarchical dense networks for single image super-resolution," *IEEE Access*, vol. 7, pp. 60572–60583, May 2019.

[16] W. Wei, G. Feng, Q. Zhang, D. Cui, M. Zhang, and F. Chen, "Accurate single image super-resolution using cascading dense connections," *Electron. Lett.*, vol. 55, no. 13, pp. 739–742, Jun. 2019.

[17] X. Mao, C. Shen, and Y.-B. Yang, "Image denoising using very deep fully convolutional encoder-decoder networks with symmetric skip connections," in *Proc. Adv. Neural Inf. Process. Syst.*, Mar. 2016, pp. 2802–2810.

[18] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Med. Image Comput. Comput.-Assist. Intervent.*, Nov. 2015, pp. 234–241.

[19] 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. 2016.

[20] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2808–2817.

[21] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.

[22] 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. (CVPR)*, Jun. 2016, pp. 1646–1654.

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

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

[25] 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.
[26] X. Wang, "Moving window-based double Haar wavelet transform for image processing," *IEEE Trans. Image Process.*, vol. 15, no. 9, pp. 2771–2779, Sep. 2006.
[27] N.-X. Lian, V. Zagorodnov, and Y.-P. Tan, "Edge-preserving image denoising via optimal color space projection," *IEEE Trans. Image Process.*, vol. 15, no. 9, pp. 2575–2587, Sep. 2006.
[28] K. Hirakawa and T. W. Parks, "Image denoising using total least squares," *IEEE Trans. Image Process.*, vol. 15, no. 9, pp. 2530–2742, Sep. 2006.
[29] M. Bertalmio, V. Caselles, and A. Pardo, "Movie denoising by average of warped lines," *IEEE Trans. Image Process.*, vol. 16, no. 9, pp. 2333–2347, Sep. 2007.
[30] C. Kervrann and J. Boulanger, "Optimal spatial adaptation for patch-based image denoising," *IEEE Trans. Image Process.*, vol. 15, no. 10, pp. 2866–2878, Oct. 2006.
[31] V. Jain and H. Seung, "Natural image denoising with convolutional networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Dec. 2008, pp. 769–776.
[32] X. Lan, S. Roth, D. Huttenlocher, and M. J. Black, "Efficient belief propagation with learned higher-order Markov random fields," in *Proc. Eur. Conf. Comput. Vis.*, May 2006, pp. 269–282.
[33] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2012, pp. 341–349.
[34] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.
[35] Y. Zhang, H. Lin, Y. Li, and H. Ma, "A patch based denoising method using deep convolutional neural network for seismic image," *IEEE Access*, vol. 7, pp. 156883–156894, Oct. 2019.
[36] F. Hashimoto, H. Ohba, K. Ote, A. Teramoto, and H. Tsukada, "Dynamic PET image denoising using deep convolutional neural networks without prior training datasets," *IEEE Access*, vol. 7, pp. 96594–96603, Jul. 2019.
[37] W. Shan, P. Liu, L. Mu, C. Cao, and G. He, "Hyperspectral image denoising with dual deep CNN," *IEEE Access*, vol. 7, pp. 171297–171312, Nov. 2019.
[38] W. Bae, J. Yoo, and J. C. Ye, "Beyond deep residual learning for image restoration: Persistent homology-guided manifold simplification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1141–1149.
[39] T. Guo, H. S. Mousavi, T. H. Vu, and V. Monga, "Deep wavelet prediction for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1100–1109.
[40] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[41] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L.-A. Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proc. Brit. Mach. Vis. Conf.*, 2012, pp. 13501–13510.
[42] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proc. 7th Int. Conf. Curves Surf.*, Jun. 2012, pp. 711–730.
[43] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th IEEE Int. Conf. Comput. Vis. (ICCV)*, Jul. 2001, pp. 416–423.
[44] 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)*, Jun. 2015, pp. 5197–5206.
[45] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. 23rd ACM Int. Conf. Multimedia (MM)*, 2015, pp. 689–692.

SHUO-FEI WANG received the B.Sc. degree in physics from the Beijing Institute of Technology, China, in 2018, where he is currently pursuing the Ph.D. degree with the School of Physics. His research interests include machine learning, image denoising, inverse problems in computer vision, and computational imaging.

WEN-KAI YU received the Ph.D. degree in computer applied technology from the University of Chinese Academy of Sciences, China, in 2015. He is currently an Associate Researcher with the School of Physics, Beijing Institute of Technology, China. His research interests include computer vision, compressed sensing, computational imaging, image reconstruction, image processing, image denoising, machine learning, and intelligent signal acquisition.

YA-XIN LI received the B.Sc. degree in physics from Northeast Normal University, China, in 2018. She is currently pursuing the M.Sc. degree with the School of Physics, Beijing Institute of Technology, China. Her research interests include single-pixel imaging, image processing, image reconstruction, image denoising, and optical communication.