Progressive Training of Multi-level Wavelet Residual Networks for Image Denoising

Yali Peng, Yue Cao, Shigang Liu, Jian Yang, and Wangmeng Zuo, Senior Member, IEEE

Abstract—Recent years have witnessed the great success of deep convolutional neural networks (CNNs) in image denoising. Albeit deeper network and larger model capacity generally benefit performance, it remains a challenging practical issue to train a very deep image denoising network. Using multi-level wavelet-CNN (MWCNN) as an example, we empirically find that the denoising performance cannot be significantly improved by either increasing wavelet decomposition levels or increasing convolution layers within each level. To cope with this issue, this paper presents a multi-level wavelet residual network (MWRN) architecture as well as a progressive training (PT-MWRN) scheme to improve image denoising performance. In contrast to MWCNN, our MWRN introduces several residual blocks after each level of discrete wavelet transform (DWT) and before inverse discrete wavelet transform (IDWT). For easing the training difficulty, scale-specific loss is applied to each level of MWRN by requiring the intermediate output to approximate the corresponding wavelet subbands of ground-truth clean image. To ensure the effectiveness of scale-specific loss, we also take the wavelet subbands of noisy image as the input to each scale of the encoder. Furthermore, progressive training scheme is adopted for better learning of MWRN by beginning with training the lowest level of MWRN and progressively training the upper levels to bring more fine details to denoising results. Experiments on both synthetic and real-world noisy images show that our PT-MWRN performs favorably against the state-of-the-art denoising methods in terms both quantitative metrics and visual quality.

Index Terms—Image denoising, convolutional neural network, wavelet transform, real-world noisy photography.

I. INTRODUCTION

Image denoising is a fundamental yet active task in image processing and low level vision with many real-world applications. On the one hand, a great number of methods [1],[2],[3] have been developed for handling synthetic noise such as additive white Gaussian noise (AWGN) with known noise variance $\sigma^2$. With the advent of deep learning, convolutional neural networks (CNNs) [5],[6],[7] have shown significant promise for image denoising and remarkably boosted the denoising performance. On the other hand, real-world image noise is generally much more complex in contrast to synthetic noise. In the recent few years, several attempts have been made to develop CNN denoisers for handling real-world noisy photographs by either modeling sophisticated real noise [8],[9], acquiring noisy and nearly noise-free image pairs [10],[11], or learning denoising models in an unsupervised or self-supervised manner [12],[13],[14].

Albeit remarkable progress has been made, it remains a challenging issue to train a high-performance deep denoising network. One direct solution for boosting denoising performance is to enlarge deep network capacity. However, simply increasing network depth and width cannot always benefit the performance improvement. For example, Table I lists the results of several variants of multi-level wavelet-CNN (MWCNN) [7] by increasing wavelet decomposition levels, convolution layers in each level, and network width. In comparison with the original MWCNN [7], the quantitative gains by MWCNN variants with more decomposition levels and more convolutional layers are very limited, i.e., at most 0.03 dB on PSNR on the BSD68 [15] when $\sigma = 25$. On the other hand, 0.04 dB gain on BSD68 can be attained by increasing the network width to its 1.5 times, but further increasing network width only brings 0.01 dB gain. Thus, it remains a challenging issue to train a deep denoising network with high model capacity, and the denoising performance cannot be significantly improved by increasing network depth and width.

In this paper, we develop a deeper and effective network, i.e., multi-level wavelet residual network (MWRN), by improving MWCNN in terms of network architecture, loss functions and training scheme. In MWCNN [7], four convolutional layers are deployed after each level of wavelet transform (DWT) and before inverse discrete wavelet transform (IDWT). However, simply stacking more convolutional layers suffers from the performance degradation problem [16]. That is, along with the increase of network depth, even the training error may stop to decrease or begin to increase. Following [16], we substitute the last three convolutional layers in each level (i.e., scale) of MWCNN with several residual blocks (i.e., four in this work), resulting in our MWRN architecture.

The introduction of residual blocks, however, cannot completely address the training difficulty issue. In [17], Lee et al. showed that the deployment of intermediate supervision on hidden layers is effective in improving the model transparency and easing training difficulty. Considering the multi-scale characteristics of MWRN, it is natural to constrain the intermediate output for each level of MWRN to approximate the...
TABLE I

| Parameters | Set12 PSNR (dB) | BSD68 PSNR (dB) | Urban100 PSNR (dB) |
|-----------|----------------|----------------|-------------------|
| (D, C, ρ) |                |                |                   |
| (3,4,1)  | 30.79          | 29.41          | 30.66             |
| (4,4,1)  | 30.82          | 29.42          | 30.73             |
| (5,4,1)  | 30.83          | 29.42          | 30.74             |
| (3,7,1)  | 30.84          | 29.44          | 30.77             |
| (3,9,1)  | 30.80          | 29.42          | 30.67             |
| (3,4,1.5)| 30.86          | 29.45          | 30.80             |
| (3,4,2)  | 30.87          | 29.46          | 30.83             |
| PT-MWRN  | 30.96          | 29.51          | 31.00             |

corresponding wavelet subbands of ground-truth clean image. Therefore, we suggest to deploy scale-specific loss to each level of MWRN. Nonetheless, we empirically find that scale-specific loss benefits moderately to denoising performance when we only take the noisy image as input to the first layer of MWRN. As a remedy, for each wavelet decomposition level in the encoder, we also take the wavelet subbands of noisy image as input, and apply an extra convolutional layer to combine it with the feature map from the upper scale.

We further present a progressive training scheme for effective learning of MWRN, resulting in our PT-MWRN method. Due to the noisy image is taken as the input for each decomposition level, it is feasible to pre-train the lower levels of MWRN without the upper levels. Thus, we first train the lowest scale of MWRN and subsequently train the upper scales to bring more fine details to the denoising results. Progressive training begins with the pre-training of a shallow network. Then, the upper scale of MWRN can be gradually trained by initializing the lower scales with the pre-trained parameters. Therefore, progressive training is also beneficial to ease the training of deeper denoising network. With the introduction of MWRN, scale-specific loss and progressive training, PT-MWRN can notably boost the denoising performance of MWCNN. It can be seen from Table I that PT-MWRN achieve PSNR gains of 0.17/0.10/0.34 dB against MWCNN for removing AWGN with σ = 25 respectively on the Set12/BSD68/Urban100 datasets.

Extensive experiments are conducted to evaluate our PT-MWRN on both grayscale and color image denoising. Ablation studies show that both residual blocks, scale-specific loss and progressive training can be utilized to ease the training of MWRN, and are effective in improving denoising performance. For Gaussian denoising, PT-MWRN performs favorably against the state-of-the-art denoising methods, e.g., BM3D [4], TNRD [18], DnCNN [5], IRCNN [19], RED [6], MemNet [20], N3Net [21], NLRRN [22], MWCNN [7] for grayscale images, and CBM3D [4], IRCNN [19], CdnCNN [5], FFDNet [23], DHDN [24] and CMWCNN [7] for color images. When paired real-world noisy and clean images are available, PT-MWRN can be easily extended to handle real-world noisy images. The results on Darmstadt Noise Dataset (DND) [10] and Smartphone Image Denoising Dataset (SIDD) [11] show that our PT-MWRN achieves the state-of-the-art denoising performance on real-world noisy photographs. Generally, the main contribution of this paper can be summarized as follows:

- A deeper multi-level wavelet residual network (MWRN) is presented for effective image denoising. In comparison to MWCNN, residual blocks are introduced in each level of the encoder and decoder, and the wavelet subbands of noisy image are taken as input for each decomposition level in the encoder.
- Scale-specific loss and progressive training are further introduced to ease the training of MWRN and improve denoising performance, resulting in our PT-MWRN.
- Quantitative and qualitative results clearly show that our PT-MWRN performs favorably against the state-of-the-art methods for both Gaussian denoising on grayscale and color images and for handling real-world noisy photographs.

The remainder of this paper is organized as follows. Section II briefly surveys the progress in deep denoising models for handling AWGN and real-world noise. Section III presents the network architecture, scale-specific loss and progressive training scheme. Section IV reports the experimental results for images with both AWGN and real-world noise. Finally, Section V ends this work with several concluding remarks.

II. RELATED WORK

In this section, we present a brief review on the development of deep denoising networks for removing AWGN and real-world image noise.

A. Learning-based Gaussian Denoising

As a fundamental image processing task, image denoising has been continuously and intensively studied for decades. Many traditional methods, e.g., total variation (TV) [1], bilateral filtering [2], sparse representation [3], fields of experts (FoE) [25], and nonlocal self-similarity (NSS) [4], [26], have been proposed. Among them, BM3D [4] and WNNM [26] are two representative NSS-based methods with decent denoising performance. Model-guided discriminative learning has also been introduced to image denoising by learning the image prior model parameters as well as compact unrolled inference in a discriminative manner [27], [18], [28]. For example, Schmidt et al. [27] presented a cascade of shrinkage fields (CSF) model by incorporating half quadratic splitting (HQS) optimization with the FoE model. Furthermore, Chen et al. [18] suggested a trainable nonlinear reaction diffusion (TNRD) approach to learn truncated gradient descent inference. And Lefkimmiatis [28] incorporated the NSS-based image prior with the learning of proximal gradient-based inference.

Instead of discriminative learning on image priors and unrolled inference, deep denoising networks aim at learning a direct mapping from the noisy image to the desired clean image. Benefited from the rapid progress in deep learning,
convolutional neural networks (CNNs) have achieved very competitive performance in comparison to both traditional methods and model-guided discriminative learning [5], [6]. In particular, Zhang et al. [5] analyzed the connection between model-guided discriminative learning and deep CNNs, and incorporated residual learning [16] with batch normalization (BN) [29] to constitute a denoising convolutional network (DnCNN). While Mao et al. [6] presented a deep fully convolutional encoding-decoding network with symmetric skip connections (i.e., RED).

Driven by the advances in deep CNNs, more and more network architectures (e.g., U-Net [30], ResNet [16], and DenseNet [31]) and modules have been introduced to the design of deep denoising networks and continually improved the Gaussian denoising performance. In [32], dense connected layers and local residual learning are incorporated to form residual dense block (RDB) for building the residual dense network (RDN). By combining residual on the residual structure and channel attention, Anwar and Barnes [33] presented a real image denoising network (RIDNet). Following the non-local neural networks [34], non-local module has also been introduced to deep denoising networks. In particular, NLRN [22] incorporates non-local module with recurrent networks, while N3Net [21] relaxes the k-nearest neighbors (KNN) selection with neural nearest neighbors block.

Several recent studies have also been given to incorporate wavelet transform [35] with CNNs for boosting denoising performance. Bae et al. [36] presented a wavelet residual network to predict the clean image in wavelet domain. Liu et al. [37] suggested a multi-level wavelet CNN (MWCNN), where DWT and IDWT are respectively deployed to substitute downsampling in encoder and up-convolution in decoder. Besides, wavelet transform has also been exploited in building deep CNNs for other low level vision tasks such as single image super-resolution [37], [38], [39]. Albeit state-of-the-art denoising performance has been achieved, it remains a challenging issue to train a deeper wavelet-based denoising network (e.g., MWCNN). Therefore, we present a deeper multi-level wavelet residual network (MWRN), and incorporate scale-specific loss and progressive training to ease the model training. Moreover, albeit our PT-MWRN is suggested for improving MWCNN, the idea of scale-specific loss and progressive training may also be beneficial to the training of other multi-scale CNNs in low level vision.

B. Deep Networks for Real-world Noisy Photographs

For real-world noisy photographs, the noise model is more sophisticated than AWGN and the parameters of noise model is unknown. Consequently, deep networks for blind Gaussian denoising generally perform poorly on real-world noisy images [10]. One plausible solution is to develop realistic noise model, which is then exploited to synthesize noisy images from clean images and to train deep denoisers. Guo et al. [8] presented a realistic noise model by considering heteroscedastic Gaussian noise and in-camera signal processing (ISP) of raw pixels, and developed a convolutional blind denoising network (CBDNet). Brooks et al. [9] also adopted signal-dependent heteroscedastic Gaussian for modeling raw image noise, and suggested to first unprocess sRGB image to raw measurement and then convert the denoised raw image to sRGB space. Wei et al. [40] provided an accurate raw image noise model for CMOS sensor by considering the imaging pipeline from photons to electrons, from electrons to voltage, and from voltage to digital numbers. Instead of explicit noise modeling, learning based models have also been introduced for modeling real-world noise and ISP procedure [41], [42].

Besides, several approaches have been developed to acquire the nearly noise-free image of given real-world noisy photograph. In particular, Plötz and Roth [10] aligned and post-processed the low ISO image to obtain nearly noise-free image, while [11], [43] achieved this goal by aligning and averaging multiple noisy images. Based on the pairs of synthetic noisy-clean images and the pairs of real noisy and nearly noise-free images, deep denoising networks [8], [44], [45] can be trained for handling real-world noisy photographs. Path-Restore [45] adopted a multi-path CNN together with a pathfinder for dynamically selecting appropriate route. In NTIRE 2019 Challenge on Real Image Denoising [46], GRDN [47], DHDN [24] and DIDN [48] have won the first three places on the sRGB track. GRDN [47] incorporated grouped residual dense blocks (GRDBs) and convolutional block attention module (CBAM) [49], while DHDN [24] and DIDN [48] are based on the modified U-Net [30] architectures. In comparison to the above methods, our PT-MWRN trained using SIDD training set and synthetic images can achieve very competitive denoising results on the DND and SIDD benchmarks.

III. PROPOSED METHOD

In this section, we present a multi-level wavelet residual network (MWRN) and a progressive training scheme for improving denoising performance. To begin with, we briefly discuss the training difficulty issue of deeper MWCNN. MWRN is then provided by introducing residual blocks and adding input layer to each scale of MWCNN. Furthermore, scale-specific loss and progressive training are suggested to ease the training of MWRN, resulting in our PT-MWRN.

A. Training Difficulty of Deeper MWCNN

Wavelet transform [35] provides an effective signal decomposition with attractive time-frequency localization characteristics, while inverse wavelet transform can accurately reconstruct the original signal for the wavelet subbands. Motivated by these characteristics, MWCNN adopts the U-Net architecture, and utilizes DWT and IDWT to respectively replace the downsampling operation in encoder and the up-convolution operation in decoder. Owing to the appealing time-frequency localization of wavelet transform, MWCNN has achieved state-of-the-art denoising performance and is effective in recovering fine image details and structures. Fig. 1(a) shows the network structure of MWCNN. In general, it involves $D$ DWT and $D$ IDWT operations. After each DWT and before each IDWT operation, $C$ convolutional layers are deployed. Specifically, we have the number of convolutional layers $C = 4$ and the number of scales $D = 3$ for the default
MWCNN in Fig. 1(a) and the number of channels in each layer is also shown in the figure.

To further improve the denoising performance, one direct solution is to enlarge the model capacity of MWCNN, e.g., increasing the $C$ and $D$ values, or the number of channels. Denote by MWCNN($D, C, \rho$) an implementation of MWCNN with the number of convolutional layers $C$, the number of scales $D$. Moreover, the number of channels of MWCNN($D, C, \rho$) is $\rho$ times of that of the default MWCNN in Fig. 1(a). From Table I it can be seen that the increase of $D$ and $C$ values benefits little to denoising performance gain. While 0.04 dB gain on BSD68 can be attained by increasing $\rho$ from 1 to 1.5, further increasing $\rho$ to 2 only brings 0.01 dB gain. The results indicate that it remains a challenging issue to train deeper/wider MWCNN. One possible cause to training difficulty is gradient vanishing/exploding. In the following, we will alleviate this issue from the aspects of network architecture, loss functions, and training strategy.

B. Network Architecture of MWRN

In this subsection, we describe the structure of multi-level wavelet residual network (MWRN) by introducing residual blocks and adding more input layers to MWCNN. Fig. 1(b) illustrates the network architecture of MWRN. For each scale of MWCNN($D, C, \rho$), we replace the last $C-1$ convolutional layers in encoder and the first $C-1$ convolutional layers in decoder with $C$ residual blocks. Using the default MWCNN(3, 4, 1) as an example, the network depth of MWCNN is 24 convolutional layers, while that of MWRN is 58 layers. As explained in [16], the deployment of residual blocks is helpful to the gradient back-propagation in training, and is effective in alleviating the gradient vanishing while improving the convergence behaviors. Thus, albeit MWRN is much deeper than MWCNN, MWRN can be easier to train due to the introduction of residual blocks.

Second, we further modify MWRN by adding an input layer to each scale of the encoder. Denote by $y$ a noisy observation with the image size $h \times w$. In terms of multi-level wavelet packet transform [50], the decomposition result at the $2^{-1}$ scale can be written as,

$$\{y_1^{(-1)}, y_2^{(-1)}, y_3^{(-1)}, y_4^{(-1)}\} = DWT(y),$$

(1)

where $y_1^{(-1)}, y_2^{(-1)}, y_3^{(-1)}, y_4^{(-1)}$ are the four wavelet subbands of $y$. Subsequently, each subband image $y_i^{(-1)}$ ($i_1 = 1, 2, 3, 4$) can be further decomposed into four subbands, i.e., $y_1^{(-2)}, y_2^{(-2)}, y_3^{(-2)}, y_4^{(-2)}$. So we can obtain the 16 wavelet
to train our MWRN, also take the input to the next convolutional layer. Analogously, we formalize the wavelet transform of the first level representation of MWRN encoder. Besides, the 64 wavelet subbands \{y_{i_1,i_2}^{(-2)}\} at the 2^{−2} scale. Analogously, the 64 wavelet subbands \{y_{i_3,i_4,i_5}^{(-3)}\} at the 2^{−3} scale can also be attained. Then, we take \{y_{i_1,i_2}^{(-2)}\} as the input to the 2^{−2} scale of MWRN. As illustrated in Fig. 1(b), a 3 × 3 convolutional layer is deployed on \{y_{i_1,i_2}^{(-2)}\} to generate a feature map \(F_y^{(−2)}\) of 256 channels. Denote by \(\text{Conv}(\text{DWT}(f_y^{(-1)}))\) the wavelet transform of the first level representation of MWRN encoder. \(F_y^{(−2)}\) and \(\text{Conv}(\text{DWT}(f_y^{(-1)}))\) are then concatenated as the input to the next convolutional layer. Analogously, we also take \{y_{i_3,i_4,i_5}^{(-3)}\} as the input to the 2^{−3} scale, generate \(F_y^{(−3)}\) of 512 channels and concatenate it with the wavelet transform \(\text{Conv}(\text{DWT}(f_y^{(-2)}))\) of the second level representation of MWRN encoder.

C. Scale-specific Loss

The \(\ell_2\)-norm reconstruction loss in MWCNN is also adopted to train our MWRN,

\[
L_r = \frac{1}{2}\|\hat{x} - x\|^2, \tag{2}
\]

where \(\hat{x}\) is the denoising result by MWRN and \(x\) is the ground-truth clean image.

In order to ease the training MWRN, we further introduce scale-specific losses respectively for the 2^{−2} and 2^{−3} scales. Let \(f_{d_{dec}}^{(-2)}\) be the feature representation of the 2^{−2} scale decoder. We use IDWT to upsample \(\text{Conv}^1(f_{d_{dec}}^{(-2)})\) and combine it with the first level representation \(f_y^{(-1)}\) of MWRN encoder. Besides, another convolutional layer is also deployed on \(f_{d_{dec}}^{(-2)}\) to predict the wavelet subbands \{\(x_{i_1,i_2}^{(-2)}\)\} at the 2^{−2} scale. Taking the residual learning formulation, we define the scale-specific loss for the 2^{−2} scale as follows,

\[
L_{s_{-2}} = \frac{1}{2}\sum_{i_1,i_2}\|\text{Conv}^2(f_{d_{dec}}^{(-2)})_{i_1,i_2} + y_{i_1,i_2}^{(-2)} - x_{i_1,i_2}^{(-2)}\|^2. \tag{3}
\]

Analogously, we can define the scale-specific loss \(L_{s_{-3}}\) for the 2^{−3} scale. We note that scale-specific losses can serve as intermediate supervision on hidden layers, and thus is effective in improving the model transparency and easing training difficulty. Moreover, it is empirically found that scale-wise input of wavelet subbands is beneficial to the denoising performance, which may be explained from two aspects. On the one hand, the successive convolutional layers in the encoder may cause potential information loss, making it difficult to accurately recover the wavelet subbands of lower scales. On the other hand, the introduction of scale-wise input makes it feasible to utilize the residual learning formulation, and is helpful in improving the convergence behavior.

Finally, the reconstruction and scale-specific losses are combined to constitute our learning objective,

\[
L = L_r + \lambda(L_{s_{-2}} + L_{s_{-3}}), \tag{4}
\]

where \(\lambda\) is a tradeoff parameter.

D. Progressive Training

The multi-scale architecture and scale-specific loss also make it feasible to train MWRN in a progressive manner. In particular, we begin with training the lowest scale of MWRN, and then progressively train the upper scales of MWRN, resulting in our PT-MWRN. For the lowest scale of MWRN, we only require to train a 19-layer CNN with eight residual

Fig. 2. Illustration of the network architectures adopted in the three stages of progressive training.
blocks. When training the upper scale of MWRN, the pre-trained lower scales of the network can serve as a good initialization for easing the network training. In the following, we describe the three stages of the progressive training scheme in more details.

In the first stage, we take \( \{x_{1,i}^{-3}\} \) as the input, and use the scale-specific loss \( \mathcal{L}^{-3} \) to train the \( 2^{-3} \) scale of MWRN. As illustrated in Fig. 2(a), a \( 3 \times 3 \) convolutional layer is deployed on \( \{x_{1,i}^{-3}\} \) to generate a feature map \( F_{y}^{-3} \) of 512 channels. Due to the upper scales are not trained yet, we simply let the second level feature representation of MWRN encoder, i.e., \( \text{Conv(DWT}(f^{-3}_{2})) \), be zero. Then, \( F_{y}^{-3} \) and \( \text{Conv(DWT}(f^{-2}_{2})) \) are concatenated and fed into the next convolutional layer. After eight residual blocks, we can obtain the feature representation \( f^{-3}_{dec} \) of the \( 2^{-3} \) scale decoder.

Finally, another convolutional layer is utilized to predict the wavelet subbands \( \{x_{1,i}^{-3}\} \) in a residual learning manner, i.e., \( x_{1,i}^{-3} = \text{Conv}^{-3}(f^{-3}_{dec}) + x_{1,i}^{-3} \).

In the second stage, we take \( \{y_{1,i}^{-2}\} \) as the input to the \( 2^{-2} \) scale of MWRN, and \( \{y_{1,i}^{-3}\} \) as the input to the \( 2^{-3} \) scale. The scale-specific loss \( \mathcal{L}_{2}^{-2} + \mathcal{L}_{3}^{-3} \) is used to train the \( 2^{-2} \) scale and finetune the \( 2^{-3} \) scale of MWRN. As shown in Fig. 2(b) we utilize a \( 3 \times 3 \) convolutional layer on \( \{y_{1,i}^{-2}\} \) to generate a feature map \( F_{y}^{-2} \) of 256 channels. Analogous to the first training stage, we assign zeros to the first level feature representation of MWRN encoder, i.e., \( \text{Conv(DWT}(f^{-1}_{2})) \), in the second stage. The concatenation of \( F_{y}^{-2} \) and \( \text{Conv(DWT}(f^{-1}_{2})) \) then goes through one convolutional layer and four residual blocks to generate the encoder representation \( f^{-2}_{dec} \) at the \( 2^{-2} \) scale. Another convolutional layer is further deployed on DWT\( (f^{-2}_{2}) \), and the output \( \text{Conv(DWT}(f^{-2}_{2})) \) is concatenated with \( F_{y}^{-2} \) and is input to the \( 2^{-3} \) scale of MWRN. Moreover, a convolutional layer is also applied to the decoder representation \( f^{-3}_{dec} \) at the \( 2^{-3} \) scale, where the IDWT of the output is then taken as the input to the decoder at the \( 2^{-2} \) scale to generate the feature representation \( f^{-2}_{dec} \). And we apply another convolutional layer on \( f^{-2}_{dec} \) to predict the wavelet subbands \( \{x_{1,i}^{-2}\} \).

In the third stage, we take \( \{y_{1,i}^{-1}\} \) as the input to the \( 2^{-1} \) scale of MWRN, and use the loss \( \mathcal{L} \) to train the \( 2^{-1} \) scale and finetune the \( 2^{-2} \) and \( 2^{-3} \) scales of MWRN. As shown in Fig. 2(c) the encoder feature map \( f_{y}^{-1} \) first passes the wavelet transform and a convolutional layer is applied, which is then concatenated with \( F_{y}^{-2} \) and input to the \( 2^{-2} \) scale of MWRN. On the other hand, a convolutional layer is also applied to the decoder representation \( f^{-2}_{dec} \) at the \( 2^{-2} \) scale, where the IDWT of the output is then taken as the input to the decoder at the \( 2^{-1} \) scale to generate the feature representation \( f^{-1}_{dec} \). And a convolutional layer is deployed on \( f^{-1}_{dec} \) to predict the denoising result \( x \). At the end of training, we merge the parameters of each batch normalization into the adjacent convolution filters, and adopt a small learning rate to further finetune PT-MWRN.

Experiments are conducted on grayscale, color, and real-world noisy images to evaluate the proposed PT-MWRN. In this section, we first describe the training and testing sets used in the experiments, and provide the implementation details of PT-MWRN. Then, we compare our PT-MWRN with the state-of-the-art denoising methods respectively for grayscale, color and real-world image denoising. All the source code and pre-trained models will be publicly available at [https://github.com/happycaoyue/PT-MWRN](https://github.com/happycaoyue/PT-MWRN).

### IV. Experiments

#### A. Experimental Setting

In this subsection, we introduce the training and testing sets used in our experiments, and describe the implementation details for training our PT-MWRN models. Following [7], we use the same training set constructed by using images from three datasets, i.e., Berkeley Segmentation Dataset (BSD) [15], DIV2K [51] and Waterloo Exploration Database (WED) [52], for image denoising. Specifically, we collect 200 images from BSD, 800 images from DIV2K, and 4,744 images from WED, to constitute our training set.

The grayscale and color images in the training set are respectively used to train the grayscale and color image denoising models. For evaluating grayscale image denoising models, we adopt three testing datasets including Set12 [5], BSD68 [15], and Urban100 [53]. For color Gaussian noisy image denoising, we also use three datasets, i.e., CBSD68 [15], Kodak24 [54] and McMaster [55]. It should be noted that all these datasets have been widely adopted for the evaluation of Gaussian denoising methods and all the test images are not included in the training set.

We randomly crop \( 24 \times 6,000 \) patches with the size of \( 240 \times 240 \) from the training images. To train PT-MWRN for Gaussian denoising with known noise level, we consider three noise levels \( \sigma = 15, 25 \) and 50. We use the ADAM [50] optimizer with \( \beta_1 = 0.9, \beta_2 = 0.999 \) and \( \epsilon = 1e^{-8} \) to minimize the loss function. As to the other hyper-parameters of ADAM, the default setting is adopted. Generally, the training

### TABLE II

The setting of learning rates for our PT-MWRN. We provide the learning rates along with the epochs in each stage, and ‘-’ denotes that the network training is early stopped in this stage.

| Stage | Epoch 1~10 | Epoch 11~20 | Epoch 21~30 | Epoch 31~40 |
|-------|------------|-------------|-------------|-------------|
| 1     | 10^{-9}    | 10^{-9}     | 10^{-9}     | -           |
| 2     | 10^{-3}    | 10^{-4}     | 10^{-5}     | 10^{-6}     |
| 3     | 10^{-3}    | 10^{-4}     | 10^{-4}     | -           |

### TABLE III

Ablation studies on PT-MWRN to illustrate the effect of (i) residual blocks (RB), (ii) scale-specific input (SI), (iii) scale-specific loss (SL) and (iv) progressive training (PT).

| Methods      | #RB | SI | SL | PT | Set12 BSD68 Urban100 |
|--------------|-----|----|----|----|--------------------|
| MWCNN        | 0   |    |    |    | 30.79 29.41 30.66   |
| MWRN(2RB)    | 2   |    |    |    | 30.84 29.43 30.78   |
| MWRN(4RB)    | 4   |    |    |    | 30.86 29.45 30.82   |
| MWRN(6RB)    | 6   | ✓  |    |    | 30.87 29.45 30.84   |
| MWRN(SL)     | 4   | ✓  | ✓  |    | 30.88 29.46 30.86   |
| MWRN(SI+SL)  | 4   | ✓  | ✓  | ✓  | 30.91 29.48 30.92   |
| PT-MWRN      | 4   | ✓  | ✓  | ✓  | 30.96 29.51 30.99   |
TABLE IV

The average PSNR(dB)/SSIM values of gray image denoising for our PT-MWCNN and the competing methods with noise levels $\sigma = 15$, 25 and 50 on the datasets Set12, BSD68 and Urban100. Red, blue and cyan are utilized to indicate top 1$^{st}$, 2$^{nd}$ and 3$^{rd}$ rank, respectively.

| Dataset  | $\sigma$ | BM3D [4] | TNRD [13] | DnCNN [5] | IRCNN [15] | RED [17] | MemNet [20] | N3Net [21] | NLRN [22] | MWCNN [7] | PT-MWRN |
|----------|---------|----------|-----------|-----------|-------------|----------|--------------|-------------|-----------|-----------|---------|
| Set12    | 15      | 32.57 / 0.8952 | 32.50 / 0.8962 | 32.86 / 0.9027 | 32.77 / 0.9008 | -         | -            | 30.55 / 0.8668 | -         | 33.16 / 0.9070 | 33.15 / 0.9068 | 33.30 / 0.9110 |
|          | 25      | 29.97 / 0.8505 | 30.05 / 0.8515 | 30.44 / 0.8618 | 30.38 / 0.8603 | -         | -            | 30.30 / 0.8599 | -         | 30.79 / 0.8714 | 30.96 / 0.8717 |
|          | 50      | 26.72 / 0.7676 | 26.82 / 0.7677 | 27.18 / 0.7827 | 27.14 / 0.7804 | 27.34 / 0.7997 | 27.38 / 0.7931 | 27.43 / 0.7931 | -         | 27.64 / 0.7980 | 27.74 / 0.8056 | 27.90 / 0.8091 |
| BSD68    | 15      | 31.06 / 0.8722 | 31.22 / 0.8822 | 31.75 / 0.8904 | 31.63 / 0.8881 | -         | -            | 29.30 / 0.8377 | -         | 29.41 / 0.8331 | 29.41 / 0.8360 | 29.51 / 0.8382 |
|          | 25      | 28.57 / 0.8017 | 28.92 / 0.8148 | 29.23 / 0.8278 | 29.15 / 0.8249 | -         | -            | 29.30 / 0.8377 | -         | 29.41 / 0.8331 | 29.41 / 0.8360 | 29.51 / 0.8382 |
|          | 50      | 25.62 / 0.6689 | 25.97 / 0.7021 | 26.23 / 0.7189 | 26.19 / 0.7171 | 26.35 / 0.7243 | 26.35 / 0.7294 | 26.39 / 0.7321 | -         | 26.47 / 0.7298 | 26.53 / 0.7366 | 26.62 / 0.7399 |
| Urban100 | 15      | 32.54 / 0.8229 | 31.98 / 0.8301 | 32.47 / 0.8401 | 32.40 / 0.8384 | -         | -            | 30.19 / 0.8926 | -         | 30.94 / 0.9018 | 30.66 / 0.9026 | 31.00 / 0.9033 |
|          | 25      | 29.70 / 0.8777 | 29.29 / 0.8731 | 29.97 / 0.8792 | 29.82 / 0.8839 | -         | -            | 30.19 / 0.8926 | -         | 30.94 / 0.9018 | 30.66 / 0.9026 | 31.00 / 0.9033 |
|          | 50      | 25.94 / 0.7791 | 25.71 / 0.7756 | 26.28 / 0.7869 | 26.14 / 0.7927 | 26.48 / 0.7991 | 26.64 / 0.8024 | 26.82 / 0.8148 | -         | 27.49 / 0.8279 | 27.42 / 0.8371 | 27.75 / 0.8397 |

Fig. 3. Gaussian denoising results of the grayscale image “09” from Set12 with noise level 25.

Fig. 4. Gaussian denoising results of the grayscale image “Test003” from BSD68 with noise level 50.
PT-MWRN involves three stages, and at most 40 epochs are adopted in each stage. The learning rate is set as Table II and we adopt the mini-batch size 24 during training. Rotation or/and flip based data augmentation is used during training. We train and test our network using the MatConvNet package [57]. All the experiments are carried out in the Windows 10 and Matlab (R2017b) environment running on a PC with Xeon E3-1230 v5 CPU, Nvidia GeForce GTX 1080 Ti GPU and 32 GB RAM. And the Nvidia CUDA 8.0 and cuDNN-v5.1 are utilized to accelerate the GPU computation. It takes about two days to train a PT-MWRN model.

B. Ablation Study

Ablation study is conducted on the grayscale images to evaluate our PT-MWRN. Generally, our PT-MWRN improves MWCNN from four aspects: (i) replacing the convolution layers with residual blocks (RB), (ii) feeding the wavelet subbands of noisy image into each scale of the encoder (i.e., scale-specific input, SI), (iii) incorporating scale-specific loss (i.e., SL), and (iv) progressive training (PT). To illustrate the effect of these components, we implement seven variants of PT-MWRN, i.e., MWRN(2RB), MWRN(4RB), MWRN(6RB), MWRN(SI), MWRN(SL), MWRN(SI+SL), and the full PT-MWRN. The configuration of each variant is provided in Table III and we also take MWCNN as a baseline method in the ablation study.

Using AWGN with $\sigma = 25$ as an example, Table III lists the PSNR and SSIM results of the PT-MWRN variants on the three grayscale image datasets, i.e., Set12, BSD68, and Urban100. In contrast to MWCNN, MWRN(4RB) achieves 0.04 dB gain by PSNR on BSD68, and further increasing the number of residual blocks contributes little to performance improvement. Thus, four residual blocks are adopted in the default PT-MWRN. Moreover, the incorporation of scale-specific input and scale-specific loss, i.e., MWRN(SI+SL), can bring another 0.04 dB gain against MWRN(4RB). In comparison, the improvements by MWRN(SI) and MWRN(SL) are much smaller, i.e., only 0.01 dB, clearly indicating that scale-specific input and scale-specific loss are complementary and can be combined to boost denoising performance. Finally, progressive training can bring another 0.03 dB gain against MWRN(SI+SL). To sum up, benefited from residual blocks, scale-specific loss and progressive training, our PT-MWRN can achieve PSNR gains of 0.17, 0.10, and 0.34 dB against MWCNN for Gaussian denoising with $\sigma = 25$ respectively on the Set12, BSD68, and Urban100 datasets.

C. Experiments on Grayscale Image Denoising

For Gaussian denoising of grayscale images, we compare our PT-MWRN with nine competing denoising methods, i.e., BM3D [4], TNRD [18], DnCNN [5], IRCNN [19], RED [6], MemNet [20], N3Net [21], NLRN [22] and MWCNN [7]. Table V lists the average PSNR(dB)/SSIM results of different methods on Set12 [5], BSD68 [15], and Urban100 [53]. It can be seen that our PT-MWRN performs favorably against all the competing methods for all the noise levels and datasets. We also note that our PT-MWRN surpasses MWCNN 0.1 dB in terms of PSNR on BSD68 and near about 0.2 dB on Set12, indicating the effectiveness of incorporating deep MWRN architecture, scale-specific loss and progressive training.

Visual comparison of denoising results is further provided. Figs. 3 and 4 show the denoising results of two grayscale images respectively from Set12 [5] ($\sigma = 25$) and BSD68 [15] ($\sigma = 30$). It can be seen that some small-scale textural details are blurred or smoothed out for the results by BM3D [4], TNRD [18], DnCNN [5], IRCNN [19], RED [6], MemNet [20], N3Net [21], MWCNN [7] and NLRN [22] are effective in preserving fine details. In comparison to the competing methods, our PT-MWRN is better at recovering image details and structures, and can obtain visually more pleasing result.

D. Experiments on Color Image Denoising

In addition to grayscale image denoising, we also train the color Gaussian image denoising model. MWCNN [7] is only trained on grayscale images in the original paper, and here we retrain MWCNN for color image denoising, dubbed by CMWCNN. For color image denoising, we compare our PT-MWRN with five competing methods, i.e., BM3D [4], IRCNN [19], CDnCNN [5], FFDNet [23], DHDN [24] and CMWCNN [7]. Table V lists the average PSNR(dB)/SSIM results of different methods on the CBSD68, Kodak24 and McMaster datasets. In terms of both PSNR and SSIM, our PT-MWRN outperforms the competing methods on the three datasets. On the Kodak24 dataset, the PSNR values of our PT-MWRN are more than 0.34 dB higher than those of the second best method, i.e., CMWCNN [7], for AWGN with any noise levels. The quantitative results further demonstrate the

| Dataset  | $\sigma$ | CBM3D [4] | IRCNN [19] | CDnCNN [5] | FFDNet [23] | DHDN [24] | CMWCNN [7] | PT-MWRN |
|----------|---------|-----------|-------------|-------------|-------------|------------|------------|---------|
| Set12    | 15      | 33.52 / 0.9215 | 33.86 / 0.9283 | 33.89 / 0.9280 | 33.84 / 0.9289 | - | 34.08 / 0.9316 | 34.22 / 0.9337 |
| 25       | 30.71 / 0.8672 | 31.17 / 0.8821 | 31.23 / 0.8826 | 31.21 / 0.8831 | 31.40 / 0.8873 | 31.58 / 0.8906 | - | 34.15 / 0.9250 |
| 50       | 27.38 / 0.7862 | 27.86 / 0.7898 | 27.92 / 0.7885 | 27.96 / 0.7911 | 28.05 / 0.7951 | 28.26 / 0.8031 | 28.40 / 0.8062 | - |
| BSD68    | 15      | 32.28 / 0.9134 | 34.55 / 0.9198 | 34.47 / 0.9198 | 34.61 / 0.9217 | - | 34.38 / 0.9250 | 34.38 / 0.9297 |
| 25       | 32.78 / 0.8767 | 32.03 / 0.8764 | 31.23 / 0.8764 | 31.13 / 0.8789 | - | 32.46 / 0.8845 | 32.62 / 0.8990 | - |
| 50       | 32.86 / 0.7973 | 32.83 / 0.7915 | 32.87 / 0.7970 | 29.40 / 0.8123 | 29.37 / 0.8115 | 29.58 / 0.8162 | - | - |
| Kodak24  | 15      | 34.06 / 0.9114 | 34.57 / 0.9196 | 33.44 / 0.9035 | 34.70 / 0.9230 | - | 34.92 / 0.9261 | 35.20 / 0.9297 |
| 25       | 34.36 / 0.8889 | 32.18 / 0.8819 | 31.52 / 0.8694 | 32.37 / 0.8874 | - | 32.55 / 0.8924 | 32.89 / 0.8979 | - |
| 50       | 32.81 / 0.7911 | 28.92 / 0.8068 | 28.62 / 0.7989 | 29.19 / 0.8192 | 29.59 / 0.8248 | 29.61 / 0.8323 | 29.87 / 0.8375 | - |
| Urban100 | 15      | 33.52 / 0.9215 | 33.86 / 0.9283 | 33.89 / 0.9280 | 33.84 / 0.9289 | - | 34.08 / 0.9316 | 34.22 / 0.9337 |
| 25       | 30.71 / 0.8672 | 31.17 / 0.8821 | 31.23 / 0.8826 | 31.21 / 0.8831 | 31.40 / 0.8873 | 31.58 / 0.8906 | - | 34.15 / 0.9250 |
| 50       | 27.38 / 0.7862 | 27.86 / 0.7898 | 27.92 / 0.7885 | 27.96 / 0.7911 | 28.05 / 0.7951 | 28.26 / 0.8031 | 28.40 / 0.8062 | - |

Table V: The average PSNR(dB)/SSIM values of color image denoising for our PT-MWRN and the competing methods with noise levels $\sigma = 15, 25$ and 50 on the CBSD68, Kodak24 and McMaster datasets. Red, blue and cyan are utilized to indicate top 1st, 2nd and 3rd rank, respectively.
effectiveness of our PT-MWRN for color image denoising. Figs. 5 and 6 respectively show the denoising results of two images by different methods. And our PT-MWRN can yield cleaner images with much finer details.

E. Running Time

We test the running time of our PT-MWRN as well as the competing methods on grayscale image denoising. Table VI gives the running time for different image sizes. In particular, PT-MWRN is much faster than RED [6] and MemNet [20], but is less efficient than TNRD [18]. DnCNN [5] and MWCNN [7]. Taking the PSNR/SSIM gains against the competing methods into account, our PT-MWRN can be regarded as a suitable tradeoff between denoising performance and computational efficiency.

F. Blind Denoising of Real-world Noisy Photographs

When trained with the pairs of real-world noisy and clean images, our PT-MWRN can also be utilized for the denoising of real-world photographs. To this end, we combine the SIDD Medium Dataset [11] and synthetic realistic noisy images adopted in [8] as our training data. Concretely, we employ the noise level estimation subnetwork in CBDNet [8] and adopt PT-MWRN as the denoising subnetwork, which are then jointly trained using the training set. The DND [10] and SIDD [10] are utilized to indicate top 1st, 2nd and 3rd rank, respectively.

Table VII lists the denoising performance of the competing methods. As the clean ground-truth images are not released, all the listed results are from the official benchmark website [4]. CDnCNN-B [5] and FFDNet [23] are trained using synthetic AWGN and thus perform poorly in handling real-world noisy photographs. In term of PSNR, our PT-MWRN outperforms all the other approaches including both traditional and deep denoisers. The SSIM values of Path-Restore [45], AINDNet [62] and our PT-MWRN are comparable, and are higher than those of the other approaches.

Fig. 7 shows the denoising results of an image from the DND benchmark [10]. CDnCNN-B is overfitted to AWGN removal and performs limited for real-world noisy images. For BM3D [4] and TNRD [18], perceptible color artifacts can still be observed. And over-smoothing of small-scale details and textures also cannot be avoided for the other competing methods. In comparison, our PT-MWRN is more effective in preserving the color and fine-scale details faithfully.

SIDD. The SIDD benchmark [10] contains 320 pairs of noisy and clean images for training, and provides 1,280 noisy images with the size 256 × 256 for testing. Analogous to DND, the clean ground-truth images of the testing noisy images are not publicly available. We compare our PT-MWRN with the state-of-the-art competitive algorithms including traditional methods BM3D [4], KSV[58] and deep denoisers CDnCNN-B [5], TNRD [18], FFDNet [23], CDNet [8], PRIDNet [61], VDN [44], CycleISP [42], Path-Restore [45] and AINDNet [62]. Table VIII lists the PSNR/SSIM results of the competing methods. As the clean ground-truth images are not released, all the listed results are from the official benchmark website [4]. CDnCNN-B and FFDNet are trained using synthetic AWGN and thus perform poorly in handling real-world noisy photographs. In term of PSNR, our PT-MWRN outperforms all the other approaches including both traditional and deep denoisers. The SSIM values of Path-Restore, AINDNet and our PT-MWRN are comparable, and are higher than those of the other approaches.

Table VIII lists the PSNR/SSIM results of the competing method. In particular, the fourth column gives the reported results of DIDN [48], DHDN [24] and GRDN [47] on the 2019 NTIRE Real Image Denoising (RID) Challenge. The testing images from the 2019 NTIRE RID Challenge are same with those in the official SIDD benchmark website [4]. The testing images from the 2019 NTIRE RID Challenge are same with those in the official SIDD benchmark website [4].
Fig. 5. Denoising results of the color image “09” from the McMaster12 dataset with noise level 15.

Fig. 6. Denoising results of the color image “227092” from CBSD68 with noise level 50.
i.e., single precision (float) and integer (uint8), respectively. So the results in the fourth and fifth columns only make little difference. The results of GRDN [47], DHDN [24] and DIDN [48] are based on the code and pre-trained models given in the authors’ personal website. From Table VIII it can be seen that our PT-MWRN is very competitive in comparison to the Top 3 winner solutions and performs much better than the other competing methods.

Finally, Table IX further compares our PT-MWRN with DIDN [48], DHDN [24] and GRDN [47] on DND and SIDD in more details. DHDN [24] and DIDN [48] apply the self-ensemble method [63] to improve denoising performance, thereby making them computationally heavy. To illustrate this point, we also give the GPU running time (in milliseconds, ms) on a 256 × 256 image in the second column of Table IX. In addition to SIDD dataset [10], GRDN [47] also applies GAN as noise simulator to generate additional synthetic training images. Overall, our PT-MWRN is the most efficient in computation, and achieves the highest PSNR on DND and the third highest PSNR on SIDD, making it very competitive in handling real-world noisy photographs.

### V. Conclusion

In this paper, we presented a deep multi-level wavelet residual network (MWRN) as well as a progressive training scheme for effective image denoising. In particular, MWRN, scale-specific loss and progressive training are incorporated to make it feasible to alleviate the difficulty of training deeper denoising networks. For ensuring the effectiveness of scale-specific loss, the wavelet subbands of noisy image are also fed into each scale of the encoder. Experiments show that our PT-MWRN performs favorably against the state-of-the-

| Method      | Time[ms] | DND[dB] | SIDD[dB] | Ensemble                          | Training Set             |
|-------------|----------|---------|----------|-----------------------------------|--------------------------|
| DIDN [48]   | 221.7    | 39.62   | 39.78    | weight ensemble, flip/rotate(×8)  | SIDD [10]                |
| DHDN [24]   | 151.5    | 39.29   | 39.84    | models(×2), flip/rotate(×8)       | SIDD [10]                |
| GRDN [47]   | 126.3    | 38.70   | 39.85    | None                              | SIDD [10] + GAN Synthesis|
| PT-MWRN     | 38.6     | 39.84   | 39.80    | None                              | SIDD [10] + CBDNet [8] Synthesis |

Fig. 7. Denoising results of an real-world noisy image from the DND dataset by different methods.
art methods for both Gaussian denoising on grayscale and color images, and can be easily extended for handling real-world noisy photographs with very competitive denoising performance. Albeit our PT-MWRN is suggested for improving MWCNN for image denoising, it may also be beneficial to the training of other multi-scale CNN architectures and other image restoration tasks, which will be further studied in the future work.

REFERENCES

[1] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” Physica D: nonlinear phenomena, vol. 60, no. 1-4, pp. 259–268, 1992.

[2] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in Proceedings of the IEEE International Conference on Computer Vision, vol. 98, no. 1, 1998, p. 2.

[3] M. Elad and M. Aharon, “Image denoising via sparse and redundant representations over learned dictionaries,” IEEE Transactions on Image Processing, vol. 15, no. 12, pp. 3736–3745, 2006.

[4] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image restoration by sparse 3d transform-domain collaborative filtering,” in Image Processing: Algorithms and Systems VI, vol. 6812. International Society for Optics and Photonics, 2008, p. 681207.

[5] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, “Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising,” IEEE Transactions on Image Processing, vol. 26, no. 7, pp. 3142–3155, 2016.

[6] X. Mao, C. Shen, and Y.-B. Yang, “Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections,” in Advances in Neural Information Processing Systems, 2016, pp. 2802–2810.

[7] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, “Multi-level wavelet-cnn for image restoration,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 773–782.

[8] S. Guo, Z. Yan, K. Zhang, W. Zuo, and L. Zhang, “Toward convolutional blind denoising of real photographs,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019.

[9] T. Brooks, B. Mildenhall, T. Xue, J. Chen, D. Sharlet, and J. T. Barron, “Unprocessing images for learned raw denoising,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 11036–11045.

[10] T. Plotz and S. Roth, “Benchmarking denoising algorithms with real photographs,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1586–1595.

[11] A. Abdelhamed, S. Lin, and M. S. Brown, “A high-quality denoising dataset for smartphone cameras,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 2018.

[12] A. Krull, T.-O. Buchholz, and F. Jug, “Noise2void-learning denoising from single noisy images,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2129–2137.

[13] S. Laine, T. Karras, J. Lehtinen, and T. Aila, “High-quality self-supervised deep image denoising,” in Advances in Neural Information Processing Systems. Curran Associates, Inc., 2019, pp. 6970–6980.

[14] X. Wu, M. Liu, Y. Cao, D. Ren, and W. Zuo, “Unpaired learning of deep image denoising,” in Proceedings of the European Conference on Computer Vision, 2020.

[15] D. Martin, C. Fowlkes, D. Tal, J. Malik et al., “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in Proceedings of the IEEE Conference on International Conference on Computer Vision, 2001, pp. 416–423.

[16] K. He, X. Zhang, S. Ren, and S. Jian, “Deep residual learning for image recognition,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[17] C. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and T. Tu, “Deeply-supervised nets,” pp. 562–570, 2015.

[18] Y. Chen and T. Pock, “Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration,” IEEE transactions on pattern analysis and machine intelligence, vol. 39, no. 6, pp. 1256–1272, 2017.

[19] K. Zhang, W. Zuo, S. Gu, and L. Zhang, “Learning deep cn denoiser prior for image restoration,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3929–3938.

[20] Y. Tai, J. Yang, X. Liu, and C. Xu, “Memnet: A persistent memory network for image restoration,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4539–4547.

[21] T. Plotz and S. Roth, “Neural nearest neighbors networks,” in Advances in Neural Information Processing Systems, 2018.

[22] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang, “Non-local recurrent network for image restoration,” in Advances in Neural Information Processing Systems, 2018, pp. 1680–1689.

[23] K. Zhang, W. Zuo, and L. Zhang, “Fidnet: Toward a fast and flexible solution for cnn-based image denoising,” IEEE Transactions on Image Processing, vol. 27, no. 9, pp. 4608–4622, 2018.

[24] B. Park, S. Yu, and J. Jeong, “Densely connected hierarchical network for image denoising,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.

[25] S. Roth and M. J. Black, “Fields of experts: A framework for learning image priors,” in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), vol. 2. Citeseer, 2005, pp. 860–867.

[26] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng, and L. Zhang, “Weighted nuclear norm minimization and its applications to low level vision,” International Journal of Computer Vision, vol. 121, no. 2, pp. 183–208, 2017.

[27] U. Schmidt and S. Roth, “Shrinkage fields for effective image restoration,” pp. 2774–2781, 2014.

[28] S. Lefkimmiatis, “Non-local color image denoising with convolutional neural networks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5892–5891.

[29] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” arXiv: Learning, 2015.

[30] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in International Conference on Medical Image computing and computer-assisted intervention. Springer, 2015, pp. 234–241.

[31] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[32] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, “Residual dense network for image super-resolution,” in CVPR, 2018.

[33] S. Anwar and N. Barnes, “Real image denoising with feature attention,” Proceedings of the IEEE International Conference on Computer Vision, 2019.

[34] X. Wang, R. Girshick, A. Gupta, and K. He, “Non-local neural networks,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[35] S. Mallat, “Wavelets for a vision,” Proceedings of the IEEE, vol. 84, no. 4, pp. 604–614, 1996.

[36] W. Bae, J. Yoo, and J. Chul Ye, “Beyond deep residual learning for image restoration: Persistent homology-guided manifold simplification,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 145–153.

[37] T. Guo, H. S. Mousavi, T. H. Vu, and V. Monga, “Deep wavelet prediction for image super-resolution,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017.

[38] H. Huang, R. He, Z. Sun, and T. Tan, “Wavelet-srnet: A wavelet-based cnn for multi-scale face super resolution,” in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1689–1697.

[39] X. Deng, R. Yang, M. Xu, and P. L. Dragotti, “Wavelet domain style transfer for an effective perception-distortion tradeoff in single image super-resolution,” in Proceedings of the IEEE International Conference on Computer Vision, October 2019.

[40] K. Wei, Y. Fu, J. Yang, and H. Huang, “A physics-based noise formation model for extreme low-light raw denoising,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020.

[41] A. Abdelhamed, M. A. Brubaker, and M. S. Brown, “Noise Model for Conditional Normalizing Flows,” in Proceedings of the IEEE International Conference on Computer Vision, 2019.

[42] S. K. Syed Waqas Zamir, Aditya Arora, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao, “Cycleisp: Real image restoration via improved data synthesis,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020.

[43] M. H. Nam, Y. Hwang, Y. J. Noh, and S. J. Kim, “A holistic approach to cross-channel image noise modeling and its application to image denoising,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.
[44] Z. Yue, H. Yong, Q. Zhao, D. Meng, and L. Zhang, “Variational denoising network: Toward blind noise modeling and removal,” in Advances in Neural Information Processing Systems. Curran Associates, Inc., 2019, pp. 1690–1701.

[45] K. Yu, X. Wang, C. Dong, X. Tang, and C. C. Loy, “Path-restore: Learning network path selection for image restoration,” arXiv preprint arXiv:1904.10343, 2019.

[46] A. Abdelhamed, R. Timofte, and M. S. Brown, “Ntire 2019 challenge on real image denoising: Methods and results,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.

[47] D.-W. Kim, J. Ryun Chung, and S.-W. Jung, “Grdn: Grouped residual dense network for real image denoising and gan-based real-world noise modeling,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.

[48] S. Yu, B. Park, and J. Jeong, “Deep iterative down-up cnn for image denoising,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.

[49] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, “Cham: Convolutional block attention module,” in Proceedings of the European Conference on Computer Vision, 2018, pp. 3–19.

[50] I. Daubechies, “Ten lectures on wavelets,” Computers in Physics, vol. 6, no. 3, pp. 1671–1671, 1998.

[51] E. Agustsson and R. Timofte, “Ntire 2017 challenge on single image super-resolution: Dataset and study,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 126–135.

[52] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang, “Waterloo exploration database: New challenges for image quality assessment models,” IEEE Transactions on Image Processing, vol. 26, no. 2, pp. 1004–1016, 2017.

[53] J.-B. Huang, A. Singh, and N. Ahuja, “Single image super-resolution from transformed self-exemplars,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[54] R. Franzen, “Kodak lossless true color image suite,” source: http://r0k.us/graphics/kodak, vol. 4, 1999.

[55] L. Zhang, X. Wu, A. Buades, and X. Li, “Color demosaicking by local directional interpolation and nonlocal adaptive thresholding,” Journal of Electronic Imaging, vol. 20, no. 2, pp. 1–15, 2011.

[56] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proceedings of the International Conference for Learning Representations, 2015.

[57] A. Vedaldi and K. Lenc, “Matconvnet: Convolutional neural networks for matlab,” in Proceedings of the 23rd ACM international conference on Multimedia. ACM, 2015, pp. 689–692.

[58] M. Aharon, M. Elad, A. Bruckstein et al., “K-svd: An algorithm for designing overcomplete dictionaries for sparse representation,” IEEE Transactions on signal processing, vol. 54, no. 11, p. 4311, 2006.

[59] J. Xu, L. Zhang, D. Zhang, and X. Feng, “Multi-channel weighted nuclear norm minimization for real color image denoising,” in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1096–1104.

[60] J. Xu, L. Zhang, and D. Zhang, “A trilateral weighted sparse coding scheme for real-world image denoising,” in Proceedings of the European Conference on Computer Vision, 2018, pp. 20–36.

[61] Y. Zhao, Z. Jiang, A. Men, and G. Ju, “Pyramid real image denoising network,” arXiv preprint arXiv:1908.00275, 2019.

[62] G. Y. P. Yoonsik Kim, Jae Woong Soh and N. I. Cho, “Transfer learning from synthetic to real-noise denoising with adaptive instance normalization,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020.

[63] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, “Enhanced deep residual networks for single image super-resolution,” in Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 2017, pp. 136–144.