Signal Processing: Image Communication 118 (2023) 117011

Contents lists available at ScienceDirect

Signal Processing: Image Communication

journal homepage: www.elsevier.com/locate/image

Deep steerable pyramid wavelet network for unified JPEG compression artifact reduction

Yi Zhang [a,*], Damon M. Chandler [b], Xuanqin Mou [a]

[a] School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, 710049, China
[b] College of Information Science and Engineering, Ritsumeikan University, Shiga, 525-8577, Japan

ARTICLE INFO

Keywords:
Compression artifact reduction
Steerable pyramid wavelet transform
Quality factor
Residual learning

ABSTRACT

Although numerous methods have been proposed to remove blocking artifacts in JPEG-compressed images, one important issue not well addressed so far is the construction of a unified model that requires no prior knowledge of the JPEG encoding parameters to operate effectively on different compression-level images (grayscale/color) while occupying relatively small storage space to save and run. To address this issue, in this paper, we present a unified JPEG compression artifact reduction model called DSPW-Net, which employs (1) the deep steerable pyramid wavelet transform network for Y-channel restoration, and (2) the classic U-Net architecture for CbCr-channel restoration. To enable our model to work effectively on images with a wide range of compression levels, the quality factor (QF) related features extracted by the convolutional layers in the QF-estimation network are incorporated in the two restoration branches. Meanwhile, recursive blocks with shared parameters are utilized to drastically reduce model parameters and shared-source residual learning is employed to avoid the gradient vanishing/explosion problem in training. Extensive quantitative and qualitative results tested on various benchmark datasets demonstrate the effectiveness of our model as compared with other state-of-the-art deblocking methods.

1. Introduction

Images/videos captured by cameras have to be compressed before being used due to the limited transmission bandwidth and storage capacity. Apart from a few cases where lossless compression is adopted (e.g., medical imaging and technical drawing), the JPEG [1] and HEVC (High efficiency video coding) [2] standards are most commonly used for lossy compression. These compressed images/videos often suffer from undesired compression artifacts such as blockiness, ringing, and blurring, all of which are introduced due to degradation of higher-frequency components in the coarse quantization stage. These compression artifacts not only give rise to degraded perceptual quality of images/videos which ultimately affects the user experience, but also have a negative impact on various computer vision algorithms that take compressed images as input. Thus, an algorithm that can effectively eliminate compression artifacts in compressed images/videos is highly desired.

Admittedly, compression artifacts can occur under many image/video coding standards such as H.264/AVC [3], HEVC, VVC (Versatile video coding) [4], and many different techniques have been proposed to improve the quality of these kinds of images/videos (e.g., [5–9]). In this paper, we focus on reducing blocking/ringing artifacts in JPEG-compressed images, which is one of the most popular image formats for photographic images. Compared with restoration techniques designed to remove noise and/or blur, designing restoration techniques to combat with JPEG compression artifacts is more challenging. This is due to the fact that the non-linearity of the quantization operations makes the quantization noise non-stationary and signal dependent [10]. For example, after quantization, the boundaries between different coding blocks become discontinuous which results in blocking artifact; *banding effects* become visible in smooth regions, and ringing artifacts appear around sharp edges. Thus, existing restoration algorithms (e.g., [11–14]) that model quantization noises as signal-independent often perform less effectively on compressed images.

To remove such non-stationary and signal dependent quantization noise, a number of deblocking/soft-decoding methods have been proposed; these approaches can be roughly classified into three different types. The first type of approach is enhancement-based, which removes compression artifacts by performing filtering operations along block boundaries in the spatial and/or frequency domains. The early spatial-domain algorithms (e.g., [15,16]) usually select a filtering mode according to the characteristics of the region and then apply the corresponding spatial-adaptive filtering near block boundaries. Later on, more complicated filtering methods were developed which include

* Corresponding author.
E-mail address: yi.zhang.osu@xjtu.edu.cn (Y. Zhang).

https://doi.org/10.1016/j.image.2023.117011
Received 23 June 2022; Received in revised form 9 January 2023; Accepted 28 June 2023
Available online 29 July 2023
0923-5965/© 2023 Published by Elsevier B.V.

postfiltering in shifted windows of image blocks [17], non-linear space-variant filtering [18], adaptive nonlocal means filtering [19], adaptive bilateral filtering [20], etc. In comparison, the frequency-domain filtering approaches directly adjust discrete cosine transform (DCT) coefficients (e.g., [17,21–24]) or reapply JPEG compression to the various shifted versions of the compressed image and then average the reapplied results [25,26].

The second type of approach is restoration-based, which treats compression artifact reduction as an ill-posed inverse problem in which prior knowledge is employed to guide the restoration process. Some typical image priors include the field of experts prior [27], the low-rank prior [28–30], the quantization constraint prior [31,32], the non-local similarity [14,33], the sparse representation prior [34–38], etc. Moreover, some approaches employ more than one image prior to operate effectively. For example, the sparse representation and quantization constraint priors were used in [39–42]; the low-rank and quantization constraint priors were used in [43]; the Laplacian prior, sparsity prior, and graph-signal smoothness prior were used in [44]. However, as mentioned in [45], most of the restoration-based deblocking algorithms are time-consuming due to the complex optimization process.

The third type of approach is learning-based, which employs deep convolutional neural network (CNN) as well as the traditional image transforms, priors, and constraints to perform the JPEG artifact removal task. Some typical CNN-based models include ARCNN [46], TNRD [47], DnCNN [48], CAS-CNN [49], MemNet [50], S-Net [51], STRRN [52], QCN [53], FeCarNet [54], ARF [55], DAGL [56], the one-to-many network [57], the deep convolutional sparse coding (DCSC) network [58], the generative adversarial networks [59] based model [60–62], and so forth. There are also some dual-domain models such as D^3 [10], IDCN [63], DMCNN [64], DDCN [65], QGAC [66], MWCNN [67], and DPW-SDNet [45], which incorporate DCT and/or discrete wavelet transform (DWT) within the network design to explore the redundant information neglected by the JPEG encoder for better restoration performance. Generally, learning-based approaches have shown to be more effective than other approaches, owning to the remarkable advantage of the deep learning techniques.

Despite the various CNN-based models being developed and the promising results so far achieved, one important issue not well addressed is the need for a unified model[1] that requires no prior knowledge of the JPEG encoding parameters to operate effectively on both the grayscale and color images with a wide range of compression levels while still occupying relatively small storage space. Specifically, for CNN-based models such as ARCNN [46], TNRD [47], CAS-CNN [49], DMCNN [64], MWCNN [67], IDCN [63], STRRN [52], and FeCarNet [54] etc., the individual network was trained for specific quality factor (QF), and thus only a few compression levels were considered. Although DnCNN [48] was trained on images with QF ranging from 5 to 99, the model works on grayscale image only and the performance is relatively weak. Subsequently, to improve the restoration performance for a variety of compression levels, a two-stage framework was proposed which first predicts the QF value of the image and then employs a specific network model corresponding to that QF value to perform the restoration task (e.g., [60,61,68]). This two-stage framework indeed works competitively on more compression levels. However, it is not a unified model, meaning that multiple network models are trained for multiple QF values. The more compression levels considered, the more models are trained, and consequently it generally requires a large storage space to save and run.

In recent years, more advanced model-based approaches have been proposed (e.g., [69,70]), which treat the deblocking task as an ill-posed inverse problem, and solve the problem via deep unfolding networks. Although unified models have been developed and trained by these methods, only a few QF values (usually $\mathrm{QF} \in \{10, 20, 30, 40\}$) were considered, and thus their performances on other compression levels/ranges are not quite impressive. To the best of our knowledge, AGARNet [71] and FBCNN [72] are the two state-of-the-art deblocking methods that tackle a wide range of compression levels via single networks. However, there is still room to further improve their performances. Specifically, for AGARNet, because the gating scheme is controlled by a coarse rescaled estimated QF map, small blocking artifacts are often left over on flat regions of the restored images, as shown in Fig. 1. For FBCNN, large storage is required to save the huge amount of network parameters.

Based on the aforementioned points, the main motivation of our work is to develop a single unified lightweight network that can blindly and effectively reduce JPEG compression artifacts over a wide range of compression levels. Towards this end, we propose a Deep Steerable Pyramid Wavelet Network (DSPW-Net), which utilizes QF-related feature maps to assist image restoration in both the spatial and frequency domains. As shown in Fig. 2, our model consists of three branches: (1) a QF-related feature extraction branch; (2) a frequency-domain Y-channel restoration branch; and (3) a spatial-domain CbCr-channel restoration branch. The QF-related feature extraction branch uses a cascade of convolutional layers to extract non-linear feature maps from the Y-channel of the input image. The Y-channel restoration branch first decomposes the Y-channel of the image into three scales and eight orientations via the complex version of the steerable pyramid wavelet transform (SPWT), resulting into one highpass residual band H_0, 24 oriented bandpass subbands $\mathrm{B}_{s,o}$ ($s = 1, 2, 3$; $o = 1, 2, \ldots, 8$), and one lowpass image L_0. These subband coefficients, along with the QF-related feature maps ($C_1 \sim C_4$) and the different scales of the original input, are then passed through six recursive blocks via residual learning to predict the uncompressed subband coefficients. Finally, the network outputs (H_0', $\mathrm{B}_{s,o}'$, L_0') are converted back to the restored Y-channel image (Y') by the inverse SPWT. The CbCr-channel restoration branch takes as input the compressed Cb and Cr channel images, the QF-related feature maps, and the restored Y-channel image, to produce the restored Cb and Cr channels. It also employs recursive blocks to reduce parameters and residual learning to prevent gradient explosion. Different from the Y-channel branch, the recursive blocks used in CbCr-channel branch constitute a spatial-domain U-Net [76] architecture, which is sufficient to restore the color information of images.

Compared with existing learning-based approaches, our method has several appealing properties. First, the QF-related feature maps are embedded into the restoration network such that a unified model can be effectively built for a wide range of compression levels. These feature maps serve as important hints that guide the network to adaptively fit different compression-level images. Compared with AGARNet [71] that utilizes only a rescaled estimated QF map, direct connection of the QF-related features can increase the nonlinearity of the restoration network to learn complex mappings at different compression levels, and thus avoid small blocking artifacts left on flat regions. Second, different from MWCNN [67] and DPW-SDNet [45] that analyze images using only the first level of the wavelet subband coefficients, DSPW-Net employs the complex version of the SPWT [77] to explore the spatial correlations of images not only at multiple scales and orientations, but also in magnitude and phase. The multi-level feature concatenation strategy additionally allows for the potential fusion of image structure information from different scales for better restoration. More importantly, by setting constraints on the steerable filters, SPWT improves upon orthonormal wavelet transforms in two useful properties: (1) it is translation-invariant (i.e., the subbands are aliasing-free), and (2) it is rotation-invariant (i.e., the subbands are steerable), both of which play important roles in representing the position and orientation of the image structure, and thus are of great importance to JPEG image restoration. Finally, the recursive blocks used in both the Y-channel and CbCr-channel restoration branches can help reduce the amount of network parameters, making the model efficient to save and run.

[1] In this paper, the term "unified model" is defined as a set of the consistent network parameters that is able to perform effective restoration on images across a wide range of different compression levels.

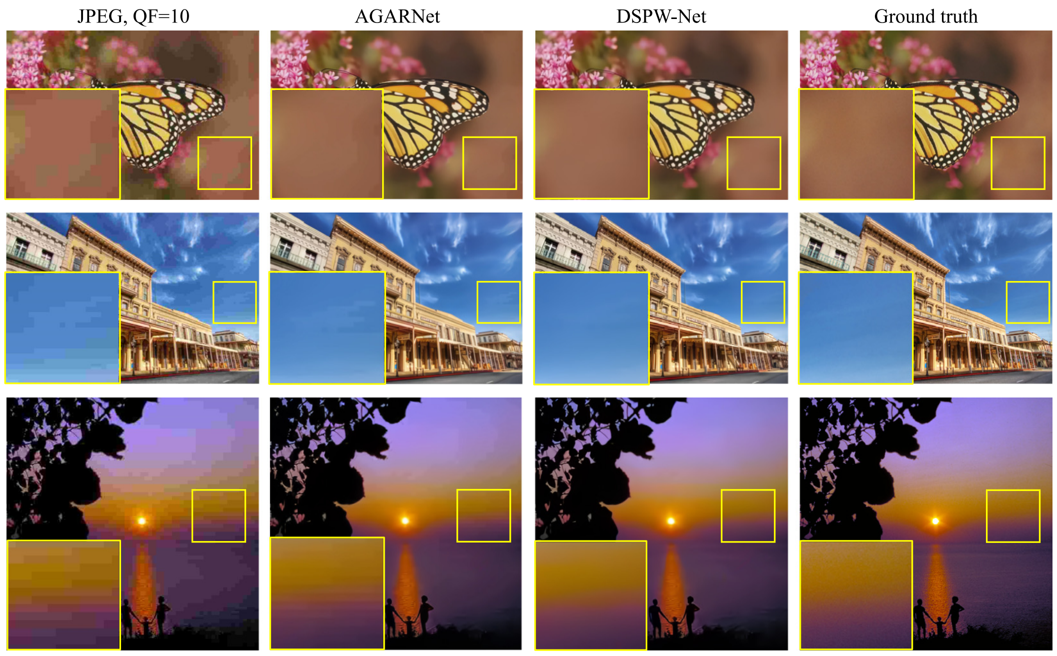

Fig. 1. An illustration of three pristine images (from the LIVE [73], Urban100 [74], and CSIQ [75] databases, respectively), their JPEG-compressed versions, and the restored images output by AGARNet [71] and DSPW-Net models, respectively. We add color to the AGARNet result by using the restored Cb/Cr channels generated by DSPW-Net. Note that small blocking artifacts can be observed on the flat regions of the AGARNet output images, while our DSPW-Net model can produce more smooth and favorable results.

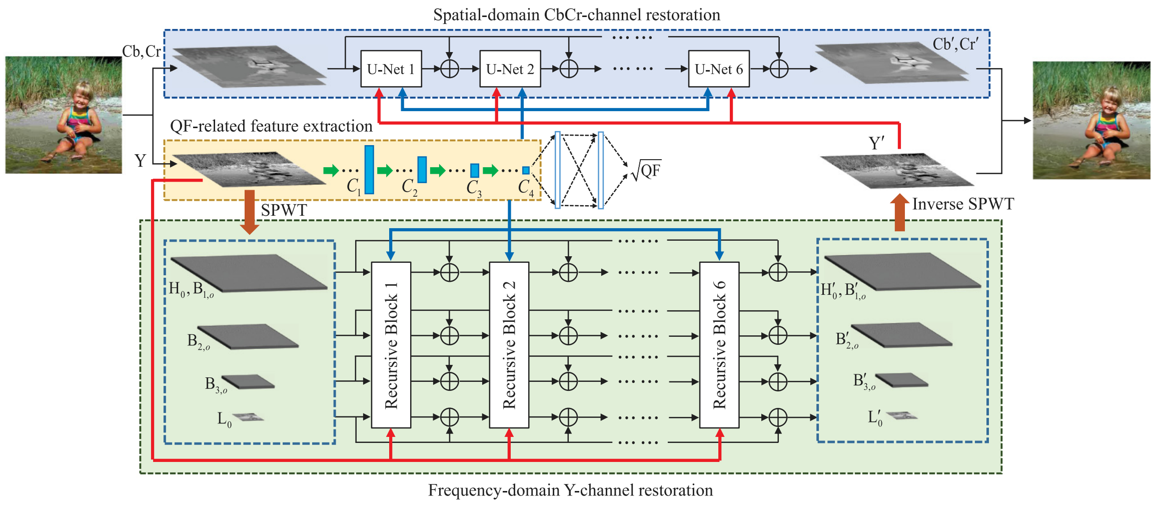

Fig. 2. A block diagram of the proposed DSPW-Net model. Detailed network architectures of the QF-related feature extraction branch, the recursive block, and the U-Net are illustrated in Figs. 3, 4, and 6, respectively.

Experimental results demonstrate the advantages of our model over other recent state-of-the-art deblocking methods.

The rest of the paper is organized as follows. Section 2 describes details of the proposed DSPW-Net. In Section 3, we analyze and discuss the performance of DSPW-Net on various JPEG-compressed images. General conclusions are presented in Section 4.

2. Algorithm

In this section, we describe the design methodology of DSPW-Net. As shown in Fig. 2, DSPW-Net takes as input an RGB compressed image and first converts it into the YCbCr space. The conversion is based on the ITU-R BT.601 standard [78] and is given by

$$\begin{cases} Y = (65.481 \cdot R + 128.553 \cdot G + 24.966 \cdot B)/255 + 16 \\ Cb = (-37.797 \cdot R - 74.203 \cdot G + 112.0 \cdot B)/255 + 128 \\ Cr = (112.0 \cdot R - 93.786 \cdot G - 18.214 \cdot B)/255 + 128 \end{cases} \tag{1}$$

where R, G, B represent the 8-bit pixel values of the three channels of the image. Then, the compressed Y and CbCr channel images are restored by the Y-channel and CbCr-channel restoration branches, respectively. To enable better performance on a wide range of compression levels, the QF-related features are incorporated in each restoration network. In the following subsections, We provide details for each branch of DSPW-Net in terms of network architecture, loss function, and network training/implementation.

2.1. QF-related feature extraction

As illustrated in Fig. 3, the QF-related feature extraction branch consists of 11 convolution layers (Conv), each of which is followed by a Rectified Linear Unit (ReLU) to extract non-linear feature maps from the Y-channel of the input image. Apart from the first and second convolution layers that contain convolution filters of kernel size 7 × 7

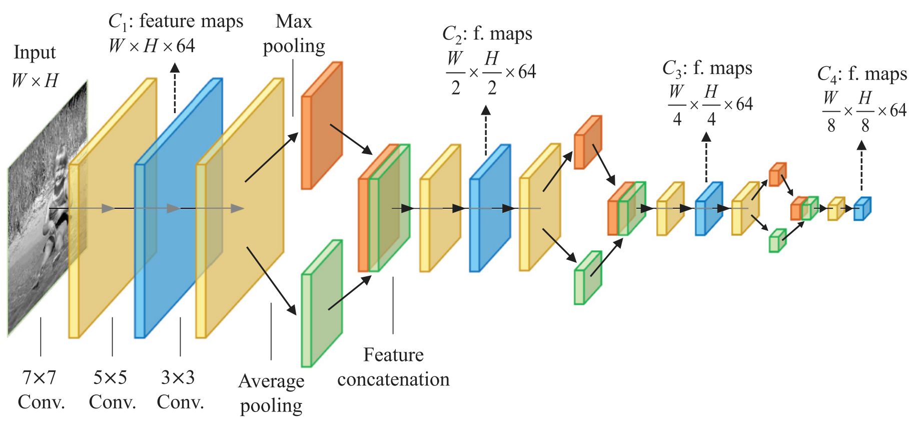

Fig. 3. Network architecture of the QF-related feature extraction branch.

and 5 × 5 pixels, respectively, to gain large receptive fields, all other convolution filters are of size 3 × 3 pixels. The network also contains three pooling layers, each of which employs both the max and average pooling strategies to extract the peak/average feature values over each non-overlapping 2 × 2-pixel region. The down-sampled feature maps obtained by applying the two pooling strategies are then concatenated in the third dimension and fed into the subsequent layers for further processing. We consistently set the channel number of the output feature maps of each convolution layer as 64, and consequently the outputs of the second, fifth, eighth, and eleventh convolution layers (marked in blue) are directly used as the QF-related features without channel rescaling.

To train the QF-related feature extraction branch, one average pooling layer and two fully connected (FC) layers have been added after C_4 such that training the QF-related feature extraction branch equals to training a QF-estimation network, and the training target is the square root of the ground-truth QF value of each image. The purpose of using $\sqrt{QF}$ instead of QF as the target value is to balance the distortion intensity variations vs. neighboring compression-level changes (see detailed explanation in Section 2.2). Given an input patch of size 128 × 128 pixels, the detailed architecture of the QF-estimation network is shown in Table 1, and we use 10,207 high-quality images from the VOC2012 database [79] for training. Specifically, after compressing each RGB image using a random QF value within the range $[10, 90]$, the non-overlapping 128 × 128 image patches were extracted from the Y-channel, and the network is optimized to minimize the error computed over the predicted and ground-truth $\sqrt{QF}$ values via the L1 loss function. In total, we extracted 63,852 patches from 10,207 compressed images. After training is done, the 11 convolution layer parameters will be fixed and used in training the two subsequent restoration networks described in the following subsections.

2.2. *Y-channel restoration*

As shown in Fig. 2, the Y-channel restoration first relies on the complex version of SPWT [77] to decompose images into multiple scales and multiple orientations, then employs recursive blocks to predict the uncompressed wavelet coefficients via residual learning, and finally transforms the wavelet coefficients back to the spatial domain through the inverse SPWT. Specifically, an input image is first split into highpass and lowpass subbands using the non-oriented highpass filter $H_0(\omega)$ and the lowpass filter $L_0(\omega)$. The lowpass subband is then decomposed into a set of oriented subbands and a lower-pass subband by using the bandpass oriented filter $B_k(\omega)$ and the narrow-band lowpass filter $L_1(\omega)$, where k denotes the index of different orientations. The lower-pass subband image can be further downsampled by a factor of 2 along both axes and decomposed again into oriented subbands and a lowpass

Table 1
Detailed network architecture of the QF-estimation network given an input patch of 128 × 128 pixel size.

Layer	Kernel size	Stride	Padding	Output size
Conv+ReLU	7 × 7	1	3	$128 \times 128 \times 64$
Conv+ReLU	5 × 5	1	2	$128 \times 128 \times 64$
Conv+ReLU	3 × 3	1	1	$128 \times 128 \times 64$
Max/Average Pooling + Concatenation				$64 \times 64 \times 128$
Conv+ReLU	3 × 3	1	1	$64 \times 64 \times 64$
Conv+ReLU	3 × 3	1	1	$64 \times 64 \times 64$
Conv+ReLU	3 × 3	1	1	$64 \times 64 \times 64$
Max/Average Pooling + Concatenation				$32 \times 32 \times 128$
Conv+ReLU	3 × 3	1	1	$32 \times 32 \times 64$
Conv+ReLU	3 × 3	1	1	$32 \times 32 \times 64$
Conv+ReLU	3 × 3	1	1	$32 \times 32 \times 64$
Max/Average Pooling + Concatenation				$16 \times 16 \times 128$
Conv+ReLU	3 × 3	1	1	$16 \times 16 \times 64$
Conv+ReLU	3 × 3	1	1	$16 \times 16 \times 64$
Average Pooling	–	–	–	1 × 64
FC	–	–	–	1 × 64
FC	–	–	–	1 × 1

residual band. Consequently, by repeatedly downsampling and decomposing the lowpass subband image, multi-scale and multi-orientation pyramid representations of the image are obtained.

For the inverse SPWT, considering a single stage of steerable pyramid decomposition, the reconstructed image in the frequency domain can be written as

$$\hat{X}(\omega) = \left\{ |H_0(\omega)|^2 + |L_0(\omega)|^2 \left(|L_1(\omega)|^2 + \sum_{k=0}^{N} |B_k(\omega)|^2\right) \right\} X(\omega) + a.t. \tag{2}$$

where N denotes the total number of the oriented filters; $a.t.$ indicates the aliasing terms. To avoid aliasing, the $L_1(\omega)$ filter is constrained to have a zero response for frequencies higher than $\pi/2$, i.e., $L_1(\omega) = 0$ for $|\omega| > \pi/2$. Also, to ensure perfect reconstruction, the transfer function of the system should be equal to one, i.e.,

$$|L_0(\omega)|^2 \left[|L_1(\omega)|^2 + \sum_{k=0}^{N} |B_k(\omega)|^2 \right] + |H_0(\omega)|^2 = 1 \tag{3}$$

The angular constraint on the bandpass filters $B_k(\omega)$ is derived from the condition of steerability [80], which can be expressed as

$$B_k(\omega) = A(\theta - \theta_k) \cdot B(\omega), \tag{4}$$

where $\theta = tan^{-1}(\omega)$, $\theta_k = \pi k/N$ for $k \in \{0, 1, \ldots, N-1\}$, and $B(\omega)$ is the radial function. Notice that $A(\theta)$ is determined by the desired derivative order, and $B(\omega)$ is constrained by both the desire to build the decomposition recursively and the need to prevent aliasing. The filters used in our work are polar-separable in the Fourier domain, and they are given by

$$L_1(r,\theta) = \begin{cases} 2\cos[\frac{\pi}{2}\log_2(\frac{4r}{\pi})], & \frac{\pi}{4} < r < \frac{\pi}{2} \\ 2 & r \le \frac{\pi}{4} \\ 0 & r \ge \frac{\pi}{2} \end{cases} \tag{5}$$

$$A(\theta-\theta_k) = \begin{cases} \alpha_k[\cos(\theta-\theta_k)]^{N-1} & |\theta-\theta_k| < \pi/2 \\ 0, & otherwise \end{cases} \tag{6}$$

$$B(r) = \begin{cases} \cos\left[\frac{\pi}{2}\log_2(\frac{2r}{\pi})\right], & \frac{\pi}{4} < r < \frac{\pi}{2} \\ 1, & r \ge \frac{\pi}{2} \\ 0, & r \le \frac{\pi}{4} \end{cases} \tag{7}$$

where r and θ are polar frequency coordinates, and

$$\alpha_k = 2^{k-1}\frac{(N-1)!}{\sqrt{N[2(N-1)]!}} \tag{8}$$

We choose $H_0(r) = B(\frac{r}{2})$ and $L_0(r,\theta) = \frac{1}{2}L_1(\frac{r}{2},\theta)$ so that the initial highpass/lowpass shape is the same as that used within the recursion. We refer interested readers to [77,81,82] for more details about the SPWT.

In this paper, we decompose an image into three pyramid levels (scales) and eight orientations, resulting into one highpass subband, three scales of bandpass subbands (each of which further contains eight bands corresponding to the eight orientations), and one lowpass subband, all of which are then fed into the recursive block for further processing. As shown in Fig. 2, the input of the recursive block contains both wavelet subband coefficients and the compressed Y-channel image rescaled to the same size as the wavelet subband, as we believe that these Y-channel images can help recover the wavelet coefficients. Because the highpass subband H_0 shares the same height and width as the first-level bandpass subband $\mathrm{B}_{1,o}$ ($o = 1,2,\ldots,8$), we concatenate them in the third dimension. Also, because the lower-pass subband L_0 is in fact a downsampled version of the original input, we do not combine it with the downsampled Y-channel image. For each of the 24 (3 scales $\times$ 8 orientations) bandpass subbands $\mathrm{B}_{s,o}$ ($s = 1,2,3$; $o = 1,2,\ldots,8$) that contains complex values, we extract their real and imaginary parts as separate feature maps. Consequently, the four-level input contains feature maps of 18 (i.e., Y-channel image + H_0 + $\Re\{\mathrm{B}_{1,o}\}$ + $\Im\{\mathrm{B}_{1,o}\}$), 17 (i.e., downsampled Y-channel image + $\Re\{\mathrm{B}_{2,o}\}$ + $\Im\{\mathrm{B}_{2,o}\}$), 17 (i.e., downsampled Y-channel image + $\Re\{\mathrm{B}_{3,o}\}$ + $\Im\{\mathrm{B}_{3,o}\}$), and 1 (i.e., L_0) channels, respectively. Here, the symbols $\Re\{\cdot\}$ and $\Im\{\cdot\}$ denote, respectively, the real and imaginary part of a complex value.

The network architecture of the recursive block is shown in Fig. 4, where C_1, C_2, C_3, and C_4 (marked in blue) denote the QF-related features obtained previously. Four paralleled branches consist of sequentially-cascaded convolution layers are applied to the four-scale subband coefficients, respectively, to predict the residual wavelet coefficients. The QF-related features are incorporated such that the network parameters can be more easily/flexibly tuned to different compression levels. Meanwhile, motivated by the U-Net [76] architecture that combines feature maps of a downsampled layer with high-resolution features to explore feature correlation among different scales, in this paper the wavelet features at different branches are also concatenated in the channel dimension to enable a potential fusion of the image structure information from different scales for better compression artifact reduction. As we will demonstrate in Section 3.3, such a fusion/concatenation technique can significantly improve the algorithm performance.

Specifically, for each branch, apart from the first and last layers that contain convolution filters of kernel size 7 × 7 and 5 × 5 pixels, respectively, all other convolution filters are of size 3 × 3 pixels. Accordingly, we use reflection paddings of 3 and 2 pixels for the first and last convolution layers, respectively, and 1 pixel zero-padding for the extra 3 × 3 convolution filters to keep the dimensions consistent between the input and output. Each convolution layer is followed by a parametric rectified linear unit (PReLU) to introduce nonlinearity to the network, and channels of the output feature maps in each layer are denoted in Fig. 4. To perform concatenation, downsampling and upsampling operations are applied such that feature maps at different levels can reach the same height and width. As illustrated in Fig. 5, we use (1) convolution filters with 1 × 1-pixel kernel size (one-pixel stride) to allow the same level feature transfer, (2) 2 × 2 average pooling operation (two-pixel stride) to downsample feature maps with scale factor 2, and (3) 2 × 2 transposed convolution (or namely deconvolution) operation (two-pixel stride) to upsample feature maps with scale factor 2. Multiple downsampling/upsampling operations are applied when larger scale factors have to be reached. For example, a double downsampling/upsampling operation can reach scale factor 4, and a triple downsampling/upsampling operation can reach scale factor 8, etc.

After concatenation, the 512-channel feature maps at each level will be fed into subsequent convolution layers to produce subband residual images, which have 17 (i.e., H_0' + $\Re\{\mathrm{B}_{1,o}'\}$ + $\Im\{\mathrm{B}_{1,o}'\}$), 16 (i.e., $\Re\{\mathrm{B}_{2,o}'\}$ + $\Im\{\mathrm{B}_{2,o}'\}$), 16 (i.e., $\Re\{\mathrm{B}_{3,o}'\}$ + $\Im\{\mathrm{B}_{3,o}'\}$), and 1 (i.e., L_0') channels, respectively. Then, these predicted residual images are combined with the original input wavelet subband using element-wise summation to generate the restored subband coefficients. As shown in Fig. 2, this shared-source skip connection is repeated for six times, and the six recursive blocks share the same network parameters to reduce model complexity. Finally, the inverse SPWT transforms the wavelet coefficients back to the spatial domain to produce the deblocked Y-channel image.

To train the Y-channel restoration branch, we built a large dataset which consists of 400 images from the training and validation set of the Berkeley Segmentation Database (BSD) [83], 850 images from the DIV2K database [84], and 4440 images from the Waterloo Exploration Database [85]. As DIV2K contains images of high resolution, both the original and down-sampled (by a scale factor of 2) versions of the 850 images in DIV2K are used as pristine images. Since our purpose is to train unified models that can effectively deal with different compression-level images, we compressed each of these 6540 images with a random QF value, which gives rise to 6540 compressed images. We also selected 100 images from the DIV2K dataset and compressed each of them with eight fixed QF values (i.e., QF∈ {10:10:80}) to further enhance the algorithm performance on some specific compression levels. Accordingly, the training dataset contains 7340 images in total. Note that when QF is small (e.g., less than 20), even ±1 change of QF value can cause significantly different distortion variation. When QF is large (e.g., more than 40), however, such distortion variation is relatively small even for large QF value changes. Thus, in this work, the random QF values were selected from the following set: QF∈ {10:20, 22:2:30, 35:5:80, 90} such that the distortion intensity variations for neighboring compression-level changes are roughly the same. In other words, the training data roughly equally distributed over different intensities of the compression distortion.

After generating 7340 JPEG-compressed images, the non-overlapping 128 × 128 patches were extracted from the Y-channel of each image. Specifically, we extracted in total 270,770 image patch pairs from both the reference and the compressed images, and the network is trained to minimize the difference between the restored image I_R and the reference image I, which is computed as a linear combination of mean square error (MSE) loss and structural similarity (SSIM) [86] loss. The pixel-wise MSE loss is defined as

$$l_{MSE} = \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}\left[I(i,j) - I_R(i,j)\right]^2, \tag{9}$$

where $I(i,j)$ and $I_R(i,j)$ denote the pixel values of spatial location (i,j) in I and I_R, respectively; W and H represent image width and height. The SSIM loss is defined as

$$l_{SSIM} = 1 - \overline{\mathrm{SSIM}\left(\mathrm{I}, \mathrm{I}_R\right)}, \tag{10}$$

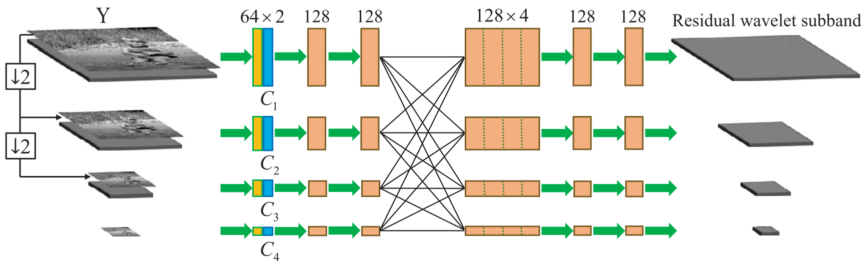

Fig. 4. Detailed network architecture of the recursive block in the Y-channel restoration branch.

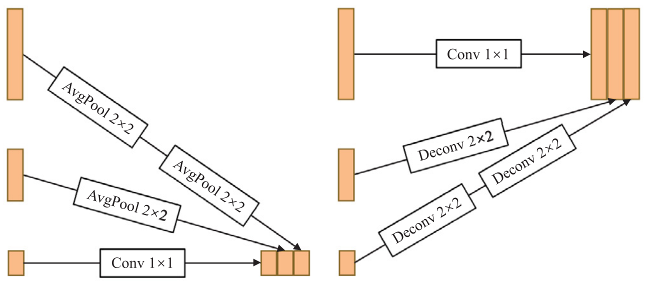

Fig. 5. An illustration of concatenating different wavelet feature maps at different levels.

where $\overline{\mathrm{SSIM}\left(\mathrm{I},\mathrm{I}_R\right)}$ denotes the average value of SSIM $\left(\mathrm{I},\mathrm{I}_R\right)$ which is computed by

$$\mathrm{SSIM}(\mathrm{I},\mathrm{I}_R) = \frac{\left(2\mu_{\mathrm{I}}\mu_{\mathrm{I}_R} + C_1\right)\left(2\sigma_{\mathrm{I}}\sigma_{\mathrm{I}_R} + C_2\right)}{\left(\mu_{\mathrm{I}}^2 + \mu_{\mathrm{I}_R}^2 + C_1\right)\left(\sigma_{\mathrm{I}}^2 + \sigma_{\mathrm{I}_R}^2 + C_2\right)}, \tag{11}$$

where $\mu_{\mathrm{I}/\mathrm{I}_R}$ and $\sigma_{\mathrm{I}/\mathrm{I}_R}$ denote, respectively, the local mean and local standard deviation of I/I_R; C_1 and C_2 are two constants which take the same values as in [86]. The overall loss function is then defined as

$$L = l_{MSE} + \lambda \cdot l_{SSIM}, \tag{12}$$

where $\lambda = 0.001$ is a parameter used to adjust the weights of the two losses.

2.3. CbCr-channel restoration

To restore the color information of JPEG-compressed images, we built a CbCr-channel restoration network within DSPW-Net model. As shown in Fig. 2, the CbCr-channel restoration branch is also a recursive network, which repeatedly utilizes the restored Y-channel image and the QF-related features to predicted the uncompressed Cb and Cr channels through residual learning. The recursive block adopts a similar U-Net architecture as illustrated in Fig. 6. The network takes as input the concatenation of Y′, Cb, and Cr channels, and outputs the residual Cb and Cr channel images. The purpose of incorporating the restored Y channel as input is to make use of the structure/texture information in luminance to guide the restoration of the color components. The QF-related features extracted in Section 2.1 are embedded in the encoder to enable the network to be adaptive to different compression levels. Since compression artifacts in the Cb and Cr channels are less perceptible, we set the outputs of all layers to have 64-channel feature maps to reduce model parameters. We set the convolution filter size to be 3 × 3 pixels for all convolution layers except the first, last, and the transposed convolution layer. As in Section 2.2, the first and last layers contain convolution filters of size 7 × 7 and 5 × 5 pixels, respectively, and the transposed convolution layers contain convolution filters of size 2 × 2 pixels. Each convolution layer is followed by PReLU to introduce nonlinearity to the network. We pad zeros around the boundaries before applying convolution to keep the sizes of all feature maps the same as the input at each scale.

To train the CbCr-channel restoration branch, the same training dataset and the same patch extraction method adopted in Section 2.2 were employed. Specifically, we extracted in total 270,770 non-overlapping 128 × 128 × 3 image patch pairs from the Y, Cb, and Cr channels of the reference and its compressed version, and the network is trained to minimize the difference between the restored Cb/Cr channel images and the reference via MSE loss defined in Eq. (9). Only MSE loss was used because (1) compression artifacts on Cb/Cr channels are less perceptible as compared with the Y channel; and (2) Cb/Cr channels typically contribute little-to-no new structure and texture information over what is already provided by the Y channel. Thus, minimizing MSE loss is sufficient to train the CbCr-channel restoration branch quite well, and consequently the computations related to SSIM loss can be saved. Note that the six U-Nets share the same network parameters to reduce model complexity. Also note that the training of CbCr-channel restoration branch requires the well-trained QF-related feature extraction model described in Section 2.1, and the well-trained Y-channel restoration model described in Section 2.2. Because of such a reliance, a small fluctuation on one network parameters during the training process may very likely cause big fluctuation on the other. After all, the update of the network parameters depends not only on the backpropagation error, but also on the changing input values. Thus, instead of jointly training the DSPW-Net model, we first trained the QF-related feature extraction branch, then the Y-channel restoration branch, and finally the CbCr-channel restoration branch. Such an independent training has the advantage of finding the optimal solution easily, because each network has a fixed input and a definite training goal/objective.

2.4. Experimental setup

All three branches in DSPW-Net were trained on a remote server using an NVIDIA GeForce RTX 3090 GPU. Since different deblocking methods may require different deep learning frameworks/environments to run, we conducted the performance test and comparison on a local workstation with an 8-core Intel i9-9900K 3.6 GHz CPU and an NVIDIA GeForce RTX 2080 SUPER GPU. The network parameters of all three branches were initialized with values sampled from a normal distribution $N(1, 0.02)$, and the leaky slopes were initialized to 0.1 for PReLU. We used the Adam algorithm [87] with an initial learning rate of 2×10^{-4} and set the exponential decay rates for the first/second moment estimate to be 0.9 and 0.999, respectively. As for other hyper-parameters of Adam, the default settings were used. For training the QF-related feature extraction branch, the learning rate was scaled down by a factor of 0.9 per epoch. We set batch size to be 8 and trained the network for 100 epochs which took about 8 h. For training the two restoration branches, the learning rate was scaled down by a factor of 0.9 after every 40,000 iterations until 10^{-7}. We set batch size to be 16 and trained the network for 25 epochs which took about two weeks.

To evaluate the performance of DSPW-Net, we used as testing data the JPEG-distorted images generated by JPEG-compressing-decompressing pristine images in five public benchmark datasets: LIVE

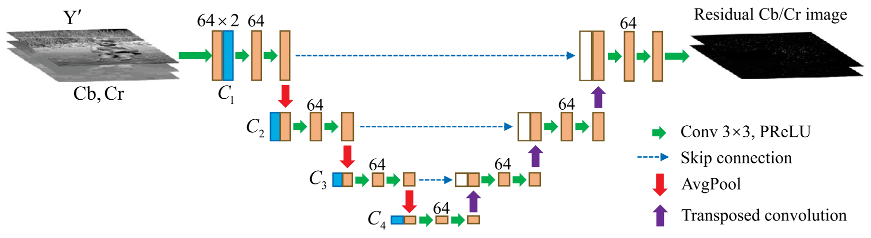

Fig. 6. Detailed network architecture of the recursive block in the CbCr-channel restoration branch.

[73], CSIQ [75], BSD100 (100 images in the validation set of BSD [83]), Urban100[2] [74], and SDIVL [88]. All images are compressed and decompressed by using the MATLAB JPEG encoder applied on RGB channels. We also conducted a test on real-compressed images by using the pristine images in the CIDIQ dataset [89]. For evaluation purposes, three criteria were used: (1) peak signal-to-noise ratio (PSNR), (2) SSIM [86], and (3) the learned perceptual image patch similarity (LPIPS) [90]. The PSNR index estimates image quality in terms of noise, while the SSIM index employs three elements (i.e., luminance, contrast, and structure) to estimate quality. LPIPS operates by measuring the “perceptual loss” between the reference and distorted images. Based on the networks used to extract deep features, the LPIPS metric can have several variants. In this paper, two LPIPS variants were adopted which employ AlexNet [91] and VGG [92] for feature extraction, and the resulting metrics are denoted by LPIPS_A and LPIPS_V, respectively. Both PSNR and SSIM were computed for the Y-channel (luminance) images only, while LPIPS was computed upon the RGB color images. Higher values for PSNR/SSIM and smaller values for LPIPS indicate more similarity between a reconstructed image and its target.

Specifically, for the restoration performance test with known compression quality factors, we compressed the reference RGB images in the four datasets (i.e., LIVE, CSIQ, BSD100, and Urban100) by using the MATLAB JPEG encoder at eight QF values: 10, 20, 30, 40, 50, 60, 70, and 80. To test the algorithm performance on images with unknown compression levels, we used the JPEG-compressed images whose QF values are less than 90 in the SDIVL dataset [88]. To specifically test the performance of DSPW-Net on images with a wide range of compression levels, we used the pristine images in SDIVL [88], and compressed each of them via the same method at QF values ranging from 10 to 90 (with a step size of 1). We also trained and tested variants of the Y-channel restoration branch which (1) does not incorporate QF-related features, (2) does not adopt multi-layer concatenation, (3) uses Laplacian pyramid/DWT instead of SPWT, (4) uses different SPWT decomposition levels, (5) uses different numbers of the recursive blocks, (6) uses different residual learning methods, and (7) uses different hyper-parameters in the loss function to illustrate the importance of some key factors in network design. Finally, we tested on CIDIQ pristine images which were first uploaded and then downloaded from the Internet to evaluate the algorithm performance on real compression.

3. Experiment results

In this section, we compare DSPW-Net with several state-of-the-art deblocking methods on benchmark datasets. We present quantitative evaluations and a qualitative comparison on images with known and unknown QF values. The performances of DSPW-Net and other QF-blind methods on a wide range of compression levels are also tested and compared. In addition, we analyze the impact of different network components, parameters, and structures on the overall restoration performance. Finally, we evaluate our method on images/photos compressed via a social media website.

[2] Due to the limited GPU memory, in our test we used as the reference image the rescaled version (three quarters of its original size) of the pristine images in the Urban100 dataset to enable CUDA acceleration.

3.1. Restoration with known QF

For performance evaluation of DSPW-Net on images with known compression levels, we generate JPEG-compressed images using eight different QF values: 10, 20, 30, 40, 50, 60, 70, and 80. Some of these values (e.g., 10 and 20, etc.) are taken into account by most of the existing deblocking algorithms. We compared DSPW-Net with 12 state-of-the-art deblocking algorithms for which code is available: (1) Pointwise Shape-Adaptive DCT [23] (SA-DCT), (2) ARCNN [46], (3) TNRD [47], (4) DnCNN [48], (5) MemNet [50], (6) MWCNN [67], (7) DMCNN [64], (8) AGARNet [71], (9) QCN [53] (10) FBCNN [72], (11) IDCN [63], and (12) MDU [70]. Note that all the learning-based algorithms except QCN, IDCN, and FBCNN were initially trained on the grayscale images only, while DSPW-Net was designed to address both grayscale and RGB color images. Also note that among these algorithms, SA-DCT, DnCNN, MemNet, AGARNet, QCN, FBCNN, and MDU consist of a single unified model trained for different compression levels, while for the others, different models were trained for different QF values.

Tables 2 and 3 present respectively the average PSNR/SSIM and LPIPS values of DSPW-Net and other deblocking algorithms tested on the aforementioned dataset images. Also included in Table 2 are the results of a baseline model (denoted by “Baseline”) whose Y-channel restoration branch was trained on images with only a few QF values i.e., QF $\in \{10, 20, 30, 40\}$. For both tables, the averaged PSNR/SSIM/LPIPS values of the original JPEG-compressed images are included for reference. The network parameter numbers, floating point operations (FLOPs; computed for a 128 × 128-pixel image), and running time (average of 30 images of size 512 × 512 pixels) of the learning-based deblocking algorithms are shown in Table 4. Among the 5.926M parameters in DSPW-Net, 0.549M are for the QF-related feature extraction branch, and 0.837M are for the CbCr-channel restoration branch. Note that ARCNN and MWCNN were originally trained on JPEG images with QF equals to 10, 20, 30, and 40 only, and thus their testing results on other QF values are not presented (similarly for TNRD, which was originally trained on images with QF $\in \{10, 20, 30\}$, and DMCNN/IDCN with QF $\in \{10, 20\}$). Accordingly, the network parameters of these QF-specific models (marked with “∗”) listed in Table 4 correspond to a single network trained for one specific QF, not all QF values. Also note that the testing results of ARCNN, TNRD, DnCNN, MemNet, MWCNN, DMCNN, AGARNet, and MDU are not presented in Table 3, as LPIPS is applied to the RGB color images only.

As can be observed in Table 2, DSPW-Net provides the best or second-best restoration performance as compared with other deblocking algorithms on all dataset images and all compression levels considered, yet with a relatively small number of network parameters. Specifically, compared with QF-specific models such as ARCNN, TNRD, DMCNN, MWCNN, and IDCN, DSPW-Net shows better performance on all dataset images. Although the performance improvements are in some cases relatively minor, DSPW-Net offers the advantage of being a single unified model for all compression levels. Compared with QF-blind single unified models such as SA-DCT, DnCNN, MemNet, AGARNet, QCN, FBCNN, and MDU, DSPW-Net shows better performance in most cases. Note that only two compression levels (i.e., QF $\in \{10, 20\}$) were originally considered in designing MemNet, and thus its

Table 2
Average PSNR and SSIM values of DSPW-Net vs. competing methods tested on various images datasets.

Dataset	Method	QF=10		QF=20		QF=30		QF=40		QF=50		QF=60		QF=70		QF=80	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
LIVE	JPEG	28.360	0.794	30.621	0.865	31.965	0.896	32.935	0.913	33.758	0.926	34.602	0.936	35.776	0.948	37.561	0.962
	SA-DCT	29.166	0.814	31.283	0.877	32.554	0.905	33.476	0.921	34.259	0.932	35.070	0.942	36.195	0.952	37.904	0.965
	ARCNN	29.409	0.821	31.731	0.884	33.127	0.913	34.085	0.927	–	–	–	–	–	–	–	–
	TNRD	26.810	0.825	27.858	0.883	28.353	0.908	–	–	–	–	–	–	–	–	–	–
	DnCNN	29.730	0.830	32.073	0.892	33.458	0.918	34.450	0.932	35.259	0.942	36.079	0.950	37.194	0.959	38.842	0.970
	MemNet	29.906	0.835	32.251	0.893	33.189	0.910	33.626	0.917	33.882	0.920	34.057	0.922	34.213	0.924	34.317	0.926
	MWCNN	30.100	0.838	32.452	0.897	33.851	0.922	34.857	0.936	–	–	–	–	–	–	–	–
	DMCNN	30.100	0.838	32.493	0.897	–	–	–	–	–	–	–	–	–	–	–	–
	AGARNet	30.034	0.835	32.455	0.897	33.891	**0.923**	34.893	0.936	35.711	**0.946**	36.531	0.953	37.620	**0.962**	39.249	**0.972**
	QCN	30.109	0.839	32.476	0.898	33.876	0.922	34.856	0.936	35.667	0.945	36.474	0.953	37.548	**0.962**	39.036	0.971
	FBCNN	30.082	0.838	32.440	0.897	33.843	0.922	34.841	0.935	35.667	0.945	36.497	0.953	37.629	**0.962**	39.283	**0.972**
	IDCN	30.046	0.838	32.415	0.897	–	–	–	–	–	–	–	–	–	–	–	–
	MDU	30.028	0.835	32.420	0.896	33.831	0.922	34.769	0.935	35.404	0.942	35.873	0.947	36.377	0.952	36.851	0.956
	Baseline	30.137	**0.840**	32.487	0.898	33.889	**0.923**	34.874	0.936	35.674	0.945	36.428	0.952	37.317	0.960	38.379	0.968
	DSPW-Net	**30.155**	**0.840**	**32.518**	**0.899**	**33.927**	**0.923**	**34.918**	**0.937**	**35.743**	**0.946**	**36.574**	**0.954**	**37.707**	**0.962**	**39.360**	**0.972**
BSD100	JPEG	28.127	0.770	30.248	0.848	31.518	0.881	32.435	0.901	33.216	0.915	34.026	0.928	35.165	0.942	36.877	0.958
	SA-DCT	28.860	0.785	30.847	0.856	32.042	0.888	32.917	0.907	33.672	0.921	34.458	0.933	35.558	0.946	37.220	0.961
	ARCNN	29.156	0.793	31.256	0.862	32.569	0.895	33.451	0.912	–	–	–	–	–	–	–	–
	TNRD	26.436	0.789	27.493	0.856	27.992	0.884	–	–	–	–	–	–	–	–	–	–
	DnCNN	29.308	0.802	31.481	0.871	32.786	0.901	33.714	0.918	34.541	0.930	35.362	0.941	36.497	0.953	38.183	0.967
	MemNet	29.467	0.805	31.648	0.870	32.581	0.891	33.032	0.899	33.336	0.904	33.543	0.908	33.746	0.911	33.929	0.914
	MWCNN	29.579	0.806	31.747	0.873	33.053	0.903	33.993	0.920	–	–	–	–	–	–	–	–
	DMCNN	29.570	0.806	31.758	0.873	–	–	–	–	–	–	–	–	–	–	–	–
	AGARNet	29.369	0.795	31.586	0.868	32.904	0.899	33.848	0.917	34.670	0.931	35.509	0.942	36.622	0.954	38.384	0.968
	QCN	29.579	0.808	31.775	0.875	33.078	0.904	34.000	0.920	34.809	**0.933**	35.620	0.943	36.727	0.955	38.307	0.968
	FBCNN	29.584	0.807	31.774	0.874	33.081	0.903	34.016	0.920	34.838	**0.933**	35.669	0.943	36.826	0.955	**38.568**	**0.969**
	IDCN	29.544	0.807	31.749	0.874	–	–	–	–	–	–	–	–	–	–	–	–
	MDU	29.550	0.804	31.764	0.873	33.059	0.904	33.944	0.919	34.598	0.929	35.112	0.936	35.682	0.943	36.304	0.950
	Baseline	29.610	**0.809**	31.789	0.875	33.090	0.904	34.017	**0.921**	34.817	**0.933**	35.579	0.943	36.533	0.954	37.738	0.965
	DSPW-Net	**29.622**	**0.809**	**31.809**	**0.876**	**33.115**	**0.905**	**34.044**	**0.921**	**34.865**	**0.933**	**35.693**	**0.944**	**36.849**	**0.956**	38.567	**0.969**
CSIQ	JPEG	28.685	0.817	31.146	0.882	32.592	0.909	33.603	0.924	34.433	0.935	35.285	0.944	36.446	0.954	38.166	0.966
	SA-DCT	29.569	0.839	31.849	0.894	33.191	0.918	34.129	0.931	34.902	0.940	35.700	0.948	36.781	0.957	38.406	0.967
	ARCNN	29.707	0.841	31.969	0.896	33.579	0.922	34.367	0.934	–	–	–	–	–	–	–	–
	TNRD	26.244	0.789	27.276	0.848	27.610	0.857	–	–	–	–	–	–	–	–	–	–
	DnCNN	30.055	0.851	32.549	0.905	34.021	0.927	35.030	0.939	35.831	0.948	36.651	0.955	37.742	0.963	39.320	0.972
	MemNet	30.102	0.854	32.396	0.904	33.514	0.919	33.863	0.924	34.166	0.927	34.278	0.929	34.416	0.931	34.531	0.932
	MWCNN	30.367	0.856	32.844	0.908	34.325	0.930	35.342	0.942	–	–	–	–	–	–	–	–
	DMCNN	30.323	0.856	32.823	0.908	–	–	–	–	–	–	–	–	–	–	–	–
	AGARNet	30.341	0.856	32.919	0.910	34.407	**0.931**	35.427	**0.943**	36.223	**0.951**	37.038	**0.958**	38.119	**0.965**	39.587	0.973
	QCN	30.410	0.858	32.932	0.910	34.384	**0.931**	35.385	0.942	36.185	0.950	36.984	0.957	38.026	**0.965**	39.432	0.973
	FBCNN	30.392	0.858	32.912	0.909	34.371	**0.931**	35.378	0.942	36.196	0.950	37.019	0.957	38.121	**0.965**	39.703	0.973
	IDCN	30.342	0.858	32.856	0.909	–	–	–	–	–	–	–	–	–	–	–	–
	MDU	30.270	0.854	32.812	0.908	34.280	0.930	35.201	0.941	35.798	0.947	36.239	0.951	36.693	0.955	37.093	0.958
	Baseline	30.452	0.859	32.942	0.910	34.393	**0.931**	35.390	0.942	36.164	0.950	36.876	0.956	37.709	0.963	38.655	0.970
	DSPW-Net	**30.468**	**0.860**	**32.974**	**0.911**	**34.435**	**0.931**	**35.435**	**0.943**	**36.247**	**0.951**	**37.061**	0.957	**38.162**	**0.965**	**39.736**	**0.974**
Urban100	JPEG	27.165	0.817	29.470	0.881	30.931	0.909	32.007	0.925	32.927	0.937	33.881	0.946	35.212	0.958	37.230	0.970
	SA-DCT	28.181	0.848	30.344	0.900	31.727	0.924	32.751	0.937	33.620	0.946	34.530	0.955	35.792	0.964	37.707	0.974
	ARCNN	28.718	0.857	31.036	0.907	32.755	0.933	33.624	0.943	–	–	–	–	–	–	–	–
	TNRD	25.813	0.843	26.793	0.889	27.270	0.908	–	–	–	–	–	–	–	–	–	–
	DnCNN	29.247	0.870	31.786	0.919	33.314	0.940	34.365	0.951	35.222	0.958	36.068	0.964	37.184	0.971	38.764	0.978
	MemNet	29.745	0.880	32.361	0.925	33.305	0.937	33.671	0.941	33.892	0.943	34.021	0.945	34.098	0.946	34.170	0.947
	MWCNN	30.355	0.890	32.969	0.932	34.533	0.949	35.644	0.959	–	–	–	–	–	–	–	–
	DMCNN	30.219	0.888	32.774	0.930	–	–	–	–	–	–	–	–	–	–	–	–
	AGARNet	30.385	0.889	33.094	0.933	34.717	0.950	35.804	**0.960**	36.678	**0.966**	37.512	**0.971**	38.545	**0.976**	40.228	**0.982**
	QCN	30.363	0.889	33.025	0.932	34.588	0.950	35.637	0.959	36.485	0.965	37.299	0.970	38.333	0.975	39.686	0.981
	FBCNN	30.195	0.886	32.853	0.930	34.440	0.948	35.530	0.958	36.418	0.964	37.286	0.970	38.431	**0.976**	40.063	**0.982**
	IDCN	30.217	0.887	32.883	0.930	–	–	–	–	–	–	–	–	–	–	–	–
	MDU	30.095	0.884	32.783	0.930	34.377	0.948	35.388	0.957	36.018	0.962	36.438	0.965	36.846	0.968	37.154	0.970
	Baseline	30.452	0.891	33.110	0.933	34.666	0.950	35.735	0.959	36.571	0.965	37.310	0.970	38.090	0.975	38.859	0.979
	DSPW-Net	**30.526**	**0.892**	**33.204**	**0.934**	**34.774**	**0.951**	**35.847**	**0.960**	**36.723**	**0.966**	**37.583**	**0.971**	**38.717**	**0.976**	**40.316**	**0.982**

Table 3
Average LPIPS values of DSPW-Net vs. competing methods tested on various images datasets.

Dataset	Method	QF=10		QF=20		QF=30		QF=40		QF=50		QF=60		QF=70		QF=80	
		LPIPS_A	LPIPS_V	LPIPS_A	LPIPS_V	LPIPS_A	LPIPS_V	LPIPS_A	LPIPS_V	LPIPS_A	LPIPS_V	LPIPS_A	LPIPS_V	LPIPS_A	LPIPS_V	LPIPS_A	LPIPS_V
LIVE	JPEG	0.1020	0.1584	0.0570	0.1057	0.0399	0.0827	0.0306	0.0696	0.0246	0.0602	0.0197	0.0522	0.0145	0.0423	0.0094	0.0305
	SA-DCT	0.1172	0.1514	0.0747	0.1122	0.0564	0.0924	0.0458	0.0801	0.0384	0.0711	0.0321	0.0631	0.0255	0.0532	0.0180	0.0411
	IDCN	0.0828	0.1292	0.0508	0.0911	–	–	–	–	–	–	–	–	–	–	–	–
	QCN	0.0806	0.1275	0.0508	0.0914	0.0383	0.0752	0.0312	0.0652	0.0260	0.0579	0.0215	0.0511	0.0168	0.0434	0.0115	0.0337
	FBCNN	0.0806	0.1281	0.0504	0.0914	0.0374	0.0747	0.0302	0.0647	0.0249	0.0572	0.0204	0.0503	0.0155	0.0422	0.0101	0.0318
	DSPW-Net	**0.0796**	**0.1263**	**0.0498**	**0.0900**	**0.0365**	**0.0734**	**0.0293**	**0.0632**	**0.0239**	**0.0557**	**0.0195**	**0.0488**	**0.0146**	**0.0408**	**0.0096**	**0.0307**
BSD100	JPEG	0.1214	0.1804	0.0773	0.1304	0.0571	0.1083	0.0480	0.0977	0.0405	0.0850	0.0312	0.0699	0.0241	0.0577	0.0170	0.0453
	SA-DCT	0.1389	0.1740	0.0982	0.1373	0.0776	0.1186	0.0681	0.1093	0.0591	0.0978	0.0498	0.0843	0.0406	0.0730	0.0297	0.0594
	IDCN	0.1036	0.1551	0.0727	0.1215	–	–	–	–	–	–	–	–	–	–	–	–
	QCN	0.1021	0.1547	0.0720	0.1221	0.0583	0.1027	0.0477	0.0909	0.0424	0.0824	0.0339	0.0706	0.0266	0.0610	0.0192	0.0495
	FBCNN	0.1024	**0.1538**	0.0710	0.1195	0.0572	**0.1020**	0.0474	0.0906	0.0408	0.0809	0.0314	0.0658	**0.0242**	**0.0545**	**0.0168**	**0.0420**
	DSPW-Net	**0.1011**	0.1556	**0.0705**	**0.1182**	**0.0553**	0.1023	**0.0463**	**0.0900**	**0.0390**	**0.0797**	**0.0298**	**0.0640**	0.0267	0.0614	0.0197	0.0496
CSIQ	JPEG	0.1201	0.1654	0.0690	0.1068	0.0468	0.0808	0.0345	0.0655	0.0266	0.0549	0.0206	0.0459	0.0147	0.0355	0.0095	0.0246
	SA-DCT	0.1424	0.1555	0.0917	0.1123	0.0670	0.0898	0.0519	0.0750	0.0417	0.0638	0.0330	0.0539	0.0242	0.0422	0.0150	0.0288
	IDCN	0.1069	0.1352	0.0656	0.0944	–	–	–	–	–	–	–	–	–	–	–	–
	QCN	0.1037	0.1330	0.0647	0.0944	0.0468	0.0755	0.0363	0.0638	0.0288	0.0546	0.0228	0.0469	0.0168	0.0378	0.0106	0.0273
	FBCNN	0.1051	0.1337	0.0653	0.0946	0.0469	0.0757	0.0364	0.0642	0.0287	0.0550	0.0228	0.0471	0.0164	0.0375	0.0099	0.0260
	DSPW-Net	**0.1025**	**0.1318**	**0.0632**	**0.0928**	**0.0450**	**0.0740**	**0.0344**	**0.0621**	**0.0270**	**0.0529**	**0.0212**	**0.0452**	**0.0151**	**0.0355**	**0.0094**	**0.0243**
Urban100	JPEG	0.1253	0.1611	0.0678	0.1108	0.0450	0.0876	0.0365	0.0754	0.0269	0.0634	0.0214	0.0543	0.0185	0.0453	0.0106	0.0314
	SA-DCT	0.1287	0.1453	0.0751	0.1036	0.0525	0.0834	0.0434	0.0730	0.0334	0.0621	0.0275	0.0539	0.0242	0.0460	0.0156	0.0331
	IDCN	0.0700	0.1012	0.0397	0.0667	–	–	–	–	–	–	–	–	–	–	–	–
	QCN	0.0689	0.0994	0.0394	0.0656	0.0292	0.0524	0.0265	0.0471	0.0198	0.0392	0.0166	0.0346	0.0166	0.0336	0.0103	0.0241
	FBCNN	0.0692	0.1013	**0.0393**	0.0670	**0.0282**	0.0525	**0.0240**	0.0459	**0.0186**	0.0388	**0.0154**	0.0339	0.0149	0.0304	**0.0086**	0.0218
	DSPW-Net	**0.0679**	**0.0976**	**0.0393**	**0.0648**	**0.0282**	**0.0508**	0.0244	**0.0452**	0.0189	**0.0378**	0.0158	**0.0331**	**0.0147**	**0.0298**	0.0092	**0.0216**

testing results on other QF values are noteworthy. The same can be said for MDU, which takes into account only four compression levels (i.e., QF $\in \{10, 20, 30, 40\}$). By comparing with the baseline model, we observe that training on more QF-value images can benefit the restoration performance, which is as expected. The better performance of "Baseline" as compared with MDU and DnCNN on some QF-value

Table 4
The network parameter numbers and running time of DSPW-Net vs. other learning-based deblocking algorithms.

Method	ARCNN* [46]	DnCNN [48]	MemNet [50]	MWCNN* [67]	DMCNN* [64]	AGARNet	QCN [53]	FBCNN [72]	IDCN* [63]	MDU [70]	DSPW-Net
# of Parameter (M)	0.106	0.669	2.095	16.170	4.700	10.370	1.434	71.922	9.900	10.490	5.925
Running time (s)	7.681	0.108	42.653	1.942	0.006	0.409	0.136	0.018	0.946	0.900	0.895
FLOPs ($\times 10^9$)	1.746	10.966	47.876	14.467	14.315	54.058	5.931	45.558	171.263	343.676	213.305

images demonstrate the advantage of the proposed multi-scale, multi-orientation SPWT network. Although AGARNet and FBCNN challenges DSPW-Net on larger QF values, in most cases AGARNet leaves small blocking artifacts in the restored images especially when QF is small (as shown in Fig. 1), and FBCNN relies on a much larger number of network parameters.

As for the LPIPS values reported in Table 3, we observe that DSPW-Net achieves better/competitive performance on most datasets especially when images are highly compressed (i.e., QF value is small). However, it is also interesting to note that when images are less compressed (i.e., QF value is large), the LPIPS value of a restored image can be larger than that of the original JPEG-compressed image, meaning that the LPIPS metric might be less effective for high-QF scenarios, and thus the testing results for high QF-value images in Table 3 are noteworthy. As for the running time and FLOPs reported in Table 4, we observe that DSPW-Net also maintains an acceptable running efficiency. It runs almost at the same speed as IDCN and MDU. Note that ARCNN and MWCNN were tested on the CPU by using their original MATLAB implementations, while the others were tested on the GPU. As for MemNet, the multiple recursive modules within the network architecture and the patch-based computation manner adopted in the original Caffe-based framework ultimately leads to a relatively lower running efficiency of the algorithm. The recursive blocks also increase the computational complexity of our model. However, we believe that such a compromise is justified given the increased restoration performance, which will be later demonstrated in Section 3.3.

Figs. 7 and 8 show visual comparisons of different deblocking methods applied on two sample grayscale JPEG-compressed images generated from the rescaled pristine images in the Urban100 [74] dataset. Also included in the two figures are the ground truth for reference. As can be observed, although all of these methods are successful at reducing the blocking artifacts, DSPW-Net seems to perform better in restoring the line structures in images, which is likely attributable to the multi-orientation and multi-scale analysis fulfilled by the complex version of steerable pyramid wavelet transform. All of these facts demonstrate that our strategy of incorporating QF-related features within the deep steerable pyramid wavelet network model is an effective way to address different compression-level images; and the effectiveness of this strategy will be further discussed in Section 3.3.

3.2. *Restoration with unknown QF*

Although the previous subsection evaluated DSPW-Net's performance on images compressed with known QF values, it is also necessary to evaluate the performance on images compressed with a wide range of QF values. To this end, we used the SDIVL dataset [88] to perform two tests. First, we directly tested DSPW-Net on the JPEG images in SDIVL. Since our model was trained to handle images with QF $\in [10, 90]$, the same compression-level images in SDIVL were tested. Second, we compressed each of the 20 pristine images in SDIVL at QF values ranging from 10 to 90 (with a step size of 1) to generate 1620 images for testing. Since our model was trained using only 27 different compression levels, our objective was to examine whether or not such a training strategy can be effective for all 81 compression levels. We compared with SA-DCT, DnCNN, MemNet, AGARNet, QCN, MDU, and FBCNN, all of which contain unified models trained on different compression levels. The results of this evaluation are shown in Table 5 and Fig. 9. Note that we measured PSNR/SSIM gain which is defined as the PSNR/SSIM value difference between the compressed and restored images in Fig. 9 to better show the algorithm performance, and each curve in Fig. 9 represents the average PSNR/SSIM gain computed over the 20 images in SDIVL for different QF values. Again, the LPIPS values for DnCNN, MemNet, AGARNet, and MDU are not reported in Table 5, as LPIPS is applied to the RGB color images only.

As can be observed, DSPW-Net shows the best performance among all algorithms considered in both tests. Specifically, SA-DCT is an image-filtering-based method that does not require training, and thus performs less effectively than DnCNN and AGARNet, both of which were trained to take into account a wide range of compression levels. For MemNet, which was trained on images with only two compression levels (i.e., QF $\in \{10, 20\}$), it is not surprising to see the performance drop in Fig. 9 and the relatively low PSNR/SSIM values in Table 5. Similarly, only four compression levels (i.e., QF $\in \{10, 20, 30, 40\}$) were considered by MDU, and thus its performance drops when QF goes larger than 40. Note that the performance of AGARNet fluctuates on some QF values and drops considerably when QF value is larger than 80 (as shown in Fig. 9). This is probably due to the fact that AGARNet has not been trained on such compression-level images. In comparison, although DSPW-Net was trained for only 27 compression levels, the same high performance can be achieved for all 81 QF values, which demonstrates the effectiveness of our model in handling a wide range of compression levels. Admittedly, equally good performance is achieved by FBCNN; however, lots of network parameters have been eliminated in our model thanks to the recursive block architecture. Also, it is interesting to note that when QF increases, the performance gains of all algorithms decreases gradually, which is as expected due to the increased image quality.

Apart from the restoration test, we additionally investigate the performance of the QF-estimation network by examining how accurately these 81 QF values can be predicted. Fig. 10 shows the mean (denoted by "$\times$") and maximum-minimum prediction error bars for each QF value, and the QF value of an overall image is computed as the average of patches with top 50% local standard deviations. Observe that our QF-estimation network can predict QF values of most images quite well especially when images are highly compressed (e.g., QF < 30). However, for larger QFs, large prediction errors occur. This result is as expected, because large QF values generally produce minor compression artifacts which confuse the network. Despite these potential prediction errors, the overall performance of DSPW-Net is not apparently affected, because our framework relies on QF-related features, not the QF values, to perform the restoration task. This tolerance allows a relatively smaller number of parameters for the QF-related feature extraction branch, leading to a more lightweight network.

3.3. *Discussion and analysis*

In this subsection, we first perform ablation study to validate the contributions of different components in the proposed network. Then, we analyze the influence of different network/training parameters towards the overall restoration performance.

3.3.1. *Contributions of network components*

As mentioned previously, incorporating QF-related features within the image restoration branch applied on steerable pyramid wavelet coefficients is an effective way to solve the challenge of being able to handle different compression levels using a single model. We also claimed in Section 2.2 that concatenating wavelet features at different levels can significantly improve the algorithm performance. To demonstrate this assertion, we trained four variant models under four

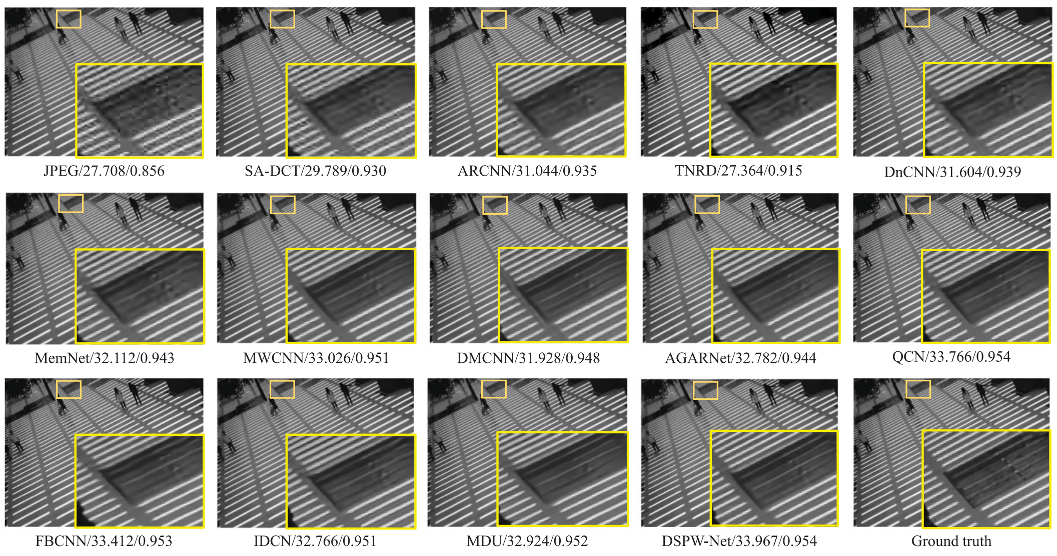

Fig. 7. Visual comparison of various deblocking methods applied on image *img_093* from the Urban100 dataset [74] compressed with QF equals to 10. The corresponding PSNR and SSIM values are presented at the bottom of each image. Notice that DSPW-Net can reproduce the steel pipe of the drainage groove more clearly as compared to the other methods.

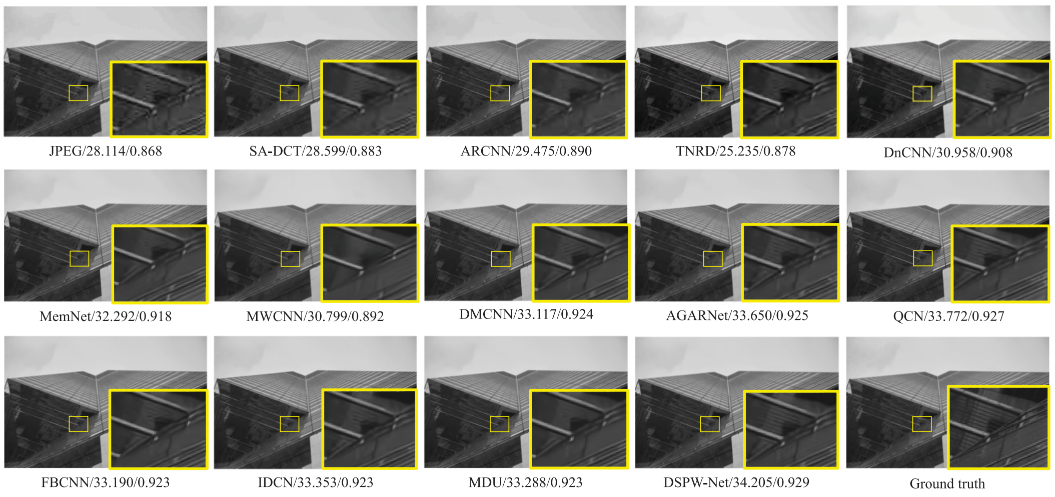

Fig. 8. Visual comparison of various deblocking methods applied on image *img_059* from the Urban100 dataset [74] compressed with QF equals to 20. The corresponding PSNR and SSIM values are presented at the bottom of each image. Notice that DSPW-Net performs particularly well at recreating the stripes reflected on the mirror wall of the building as compared to the other methods.

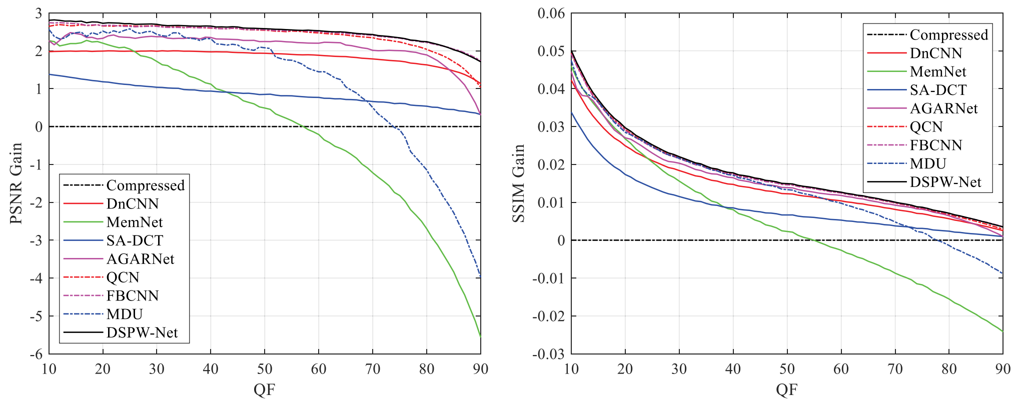

Fig. 9. Average PSNR and SSIM gains obtained by applying SA-DCT, DnCNN, MemNet, AGARNet, QCN, FBCNN, MDU, and DSPW-Net on JPEG-compressed images generated from the pristine images in the SDIVL dataset [88]. For each figure, the x-axis represents the QF of the input and y-axis represents the corresponding performance gain.

Table 5
Average PSNR and SSIM values of DSPW-Net vs. competing methods tested on the SDIVL dataset [88].

	JPEG	SA-DCT [23]	DnCNN [48]	MemNet [50]	AGARNet [71]	QCN [53]	FBCNN [72]	MDU [70]	DSPW-Net
PSNR	37.903	38.721	39.698	37.881	39.958	40.229	40.296	38.938	**40.321**
SSIM	0.948	0.958	0.964	0.954	0.965	0.966	0.966	0.963	**0.967**
LPIPS_A	0.0353	0.0445	–	–	–	0.0275	0.0270	–	**0.0263**
LPIPS_V	0.0635	0.0655	–	–	–	0.0470	0.0466	–	**0.0455**

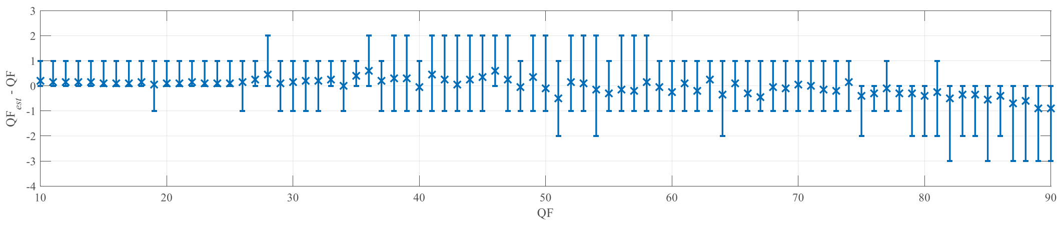

Fig. 10. Mean QF prediction errors and maximum-minimum error bars tested on SDIVL [88] dataset images compressed with QF ∈ {10 : 90}. Note that the symbol "×" on each bar represents the mean QF prediction error computed over 20 images; the top and bottom sides of each bar represent the maximal and minimal QF prediction errors, respectively.

different conditions, respectively: (1) without incorporating the QF-related features (denoted by DSPW-Net$_{NF}$); (2) without using the multi-level feature concatenation (denoted by DSPW-Net$_{NC}$); (3) replacing SPWT with Laplacian pyramid (denoted by Lapyr-Net); and (4) replacing SPWT with DWT (denoted by DWT-Net). In addition, to demonstrate the effectiveness of the shared-source skip connection used in our network, we trained another three models using three different residual learning methods as illustrated in Fig. 11: (1) no skip connection; (2) one skip connection; and (3) distinct-source skip connection. Correspondingly, the three trained models are denoted by DSPW-Net$_{NS}$, DSPW-Net$_{OS}$, and DSPW-Net$_{DS}$, respectively. Note that all these models were trained by using the same training data and parameter settings. In Lapyr-Net, a Gaussian filter of 7 × 7-pixel size and a standard deviation of one was applied to generate the Laplacian pyramid. In DWT-Net, the Daubechies 3 wavelet was employed to decompose the image into three levels, and only $C_2 \sim C_4$ features were incorporated in the Y-channel restoration branch, because the height/width of the first level wavelet subband is only half of the original input. In DSPW-Net$_{NF}$, the multi-layer concatenation is replaced by 1 × 1 convolution filters that are applied to the 128-channel features at each scale outputting 512-channel features such that the overall network parameter number does not change significantly. For DSPW-Net$_{NS}$, the common strategy of gradient clipping is employed to avoid the exploding gradient problem. We tested the seven models on the same JPEG-compressed images as used in Section 3.1. The results on the LIVE, BSD100, CSIQ, and Urban100 datasets in terms of PSNR and SSIM are presented in Table 6. Also included are the results of the original DSPW-Net for comparison.

As these results demonstrate, Lapyr-Net, DWT-Net, DSPW-Net$_{NC}$, and DSPW-Net$_{NF}$ generally perform less effectively than DSPW-Net, meaning that the multi-level feature concatenation, the QF-related feature incorporation, and the steerable pyramid wavelet transform are the three key elements in the network design. The relatively weak performance of Lapyr-Net and DWT-Net seems to suggest that analyzing the steerable pyramid magnitude/phase representation of images can be more effective in JPEG-image restoration task. By comparing the three elements, we further observe that the multi-level feature concatenation applied on steerable pyramid wavelet subband seems to play much important roles in boosting the overall performance, suggesting that features of different scales/orientations in the wavelet domain have to be fused for better restoration. In contrast, the QF-related features seem to be less important which is as expected because these features only serve to assist our model in tackling different compression levels. By comparing with DSPW-Net$_{NS}$, DSPW-Net$_{DS}$, and DSPW-Net$_{OS}$, we observe that the shared-source skip connection generally performs better than other residual learning methods. In fact, this residual learning method has been used in many super-resolution networks such as DRRN [93] and LapSRN [94], and our results demonstrate that it is also suitable for JPEG compression artifact reduction.

3.3.2. *Network/training parameters*

In this subsection, we analyze the influence of some important network/training parameters towards the overall restoration performance. These important parameters are (1) the decomposition level "S" of the SWPT, (2) the recursive block number "R", and (3) the weight "λ" in the loss function. Specifically, we trained eight variant models for the restoration branch by using the same training data. Among these eight variants, two models employ a SPWT that decomposes images into one and two scales respectively; three models use respectively two, four, and eight recursive blocks; and the extra three models were trained with different λ values in Eq. (12). Note that when the number of decomposition levels of the SPWT changes, the number of QF-related features incorporated in the restoration branch also varies. For example, a two-scale decomposition (i.e., $S = 2$) will produce 16 oriented bandpass subbands, and thus only the three fine-scale QF-related features (i.e., C_1, C_2, and C_3 in Fig. 4) will be concatenated. Similarly, a one-scale decomposition (i.e., $S = 1$) will require only two QF-related features (i.e., C_1, C_2 in Fig. 4) to be concatenated. We tested the eight models on the same JPEG-compressed images as used in Section 3.1. The results on the LIVE, BSD100, CSIQ, and Urban100 datasets in terms of PSNR and SSIM are presented in Table 7. Again, the results of the original DSPW-Net are included for comparison.

As can be seen from Table 7, the performance of DSPW-Net generally improves as the number of decomposition levels and recursive blocks increases. However, such an increased performance comes at the expense of a heavier computational burden. In particular, compared with DSPW-Net, only a slight performance increase is obtained by the "$R = 8$" model. For some cases, the performance does not change or even slightly decrease. Thus, by considering the performance improvement, computational cost, and the available hardware, we adopted 6 recursive blocks in our restoration branch. As for the different combinations of the two terms in Eq. (12), we observe that larger λ values generally produce higher SSIM and lower PSNR values. This is as expected, because larger λ values indicate more emphasis being placed on minimizing the structure similarity between the restored and ground truth images by the optimization algorithm. We empirically set $\lambda = 10^{-3}$ in this work to balance the two metrics.

3.4. *Test on real compression*

The aforementioned tests were all conducted on synthetically compressed images. In this subsection, we test DSPW-Net on real-world

Table 6
Performance of seven variant models of DSPW-Net trained by using different network components. See text for more details.

Dataset	Method	QF=10		QF=20		QF=30		QF=40		QF=50		QF=60		QF=70		QF=80	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
LIVE	DSPW-Net$_{NC}$	30.053	0.838	32.425	0.898	33.834	0.922	34.826	0.936	35.654	0.945	36.480	0.953	37.614	0.962	39.271	0.972
	DSPW-Net$_{NF}$	30.136	0.839	32.503	0.898	33.904	0.923	34.895	0.936	35.722	0.946	36.548	0.953	37.679	0.962	39.322	0.972
	Lapyr-Net	30.031	0.838	32.386	0.897	33.787	0.921	34.773	0.935	35.596	0.945	36.423	0.952	37.557	0.961	39.202	0.971
	DWT-Net	29.871	0.833	32.195	0.894	33.573	0.919	34.549	0.933	35.368	0.943	36.190	0.951	37.315	0.960	38.955	0.970
	DSPW-Net$_{NS}$	30.047	0.838	32.375	0.898	33.745	0.922	34.702	0.936	35.495	0.945	36.284	0.953	37.348	0.962	38.866	0.971
	DSPW-Net$_{DS}$	30.122	0.839	32.491	0.898	33.895	0.923	34.885	0.936	35.710	0.946	36.538	0.953	37.671	0.962	39.327	0.972
	DSPW-Net$_{OS}$	30.138	0.840	32.503	0.899	33.909	0.923	34.896	0.936	35.722	0.946	36.552	0.953	37.686	0.962	39.330	0.972
	DSPW-Net	30.155	0.840	32.518	0.899	33.927	0.923	34.918	0.937	35.743	0.946	36.574	0.954	37.707	0.962	39.360	0.972
BSD100	DSPW-Net$_{NC}$	29.543	0.808	31.742	0.875	33.052	0.904	33.985	0.921	34.805	0.933	35.632	0.943	36.788	0.955	38.511	0.969
	DSPW-Net$_{NF}$	29.608	0.808	31.797	0.875	33.102	0.904	34.034	0.921	34.853	0.933	35.679	0.944	36.831	0.956	38.529	0.969
	Lapyr-Net	29.534	0.807	31.712	0.874	33.013	0.903	33.942	0.920	34.759	0.932	35.586	0.943	36.741	0.955	38.459	0.969
	DWT-Net	29.414	0.804	31.577	0.872	32.862	0.902	33.782	0.919	34.590	0.931	35.411	0.942	36.553	0.954	38.236	0.968
	DSPW-Net$_{NS}$	29.442	0.807	31.549	0.874	32.780	0.903	33.646	0.920	34.400	0.932	35.151	0.943	36.178	0.955	37.650	0.968
	DSPW-Net$_{DS}$	29.601	0.809	31.792	0.876	33.094	0.905	34.024	0.921	34.844	0.933	35.672	0.944	36.827	0.956	38.545	0.969
	DSPW-Net$_{OS}$	29.613	0.809	31.800	0.876	33.104	0.905	34.036	0.921	34.855	0.933	35.682	0.944	36.838	0.956	38.541	0.969
	DSPW-Net	29.622	0.809	31.809	0.876	33.115	0.905	34.044	0.921	34.865	0.933	35.693	0.944	36.849	0.956	38.567	0.969
CSIQ	DSPW-Net$_{NC}$	30.363	0.858	32.872	0.910	34.342	0.931	35.346	0.942	36.151	0.950	36.969	0.957	38.071	0.965	39.649	0.973
	DSPW-Net$_{NF}$	30.441	0.859	32.953	0.910	34.408	0.931	35.410	0.943	36.221	0.951	37.039	0.957	38.142	0.965	39.717	0.974
	Lapyr-Net	30.350	0.858	32.844	0.909	34.295	0.930	35.290	0.942	36.099	0.950	36.915	0.957	38.010	0.964	39.575	0.973
	DWT-Net	30.215	0.854	32.694	0.907	34.139	0.929	35.133	0.940	35.938	0.949	36.752	0.956	37.850	0.964	39.407	0.972
	DSPW-Net$_{NS}$	30.220	0.857	32.588	0.909	33.961	0.930	34.860	0.941	35.594	0.949	36.309	0.956	37.260	0.964	38.560	0.972
	DSPW-Net$_{DS}$	30.439	0.859	32.949	0.911	34.410	0.931	35.411	0.943	36.220	0.951	37.035	0.957	38.137	0.965	39.705	0.973
	DSPW-Net$_{OS}$	30.452	0.859	32.962	0.911	34.421	0.931	35.420	0.943	36.231	0.951	37.045	0.957	38.147	0.965	39.709	0.974
	DSPW-Net	30.468	0.860	32.974	0.911	34.435	0.931	35.435	0.943	36.247	0.951	37.061	0.957	38.162	0.965	39.736	0.974
Urban100	DSPW-Net$_{NC}$	30.249	0.888	32.945	0.931	34.528	0.949	35.603	0.958	36.483	0.965	37.341	0.970	38.476	0.976	40.086	0.982
	DSPW-Net$_{NF}$	30.502	0.891	33.180	0.934	34.751	0.950	35.832	0.960	36.711	0.966	37.566	0.971	38.695	0.976	40.288	0.982
	Lapyr-Net	30.116	0.885	32.784	0.930	34.346	0.948	35.422	0.957	36.304	0.964	37.171	0.969	38.319	0.975	39.938	0.982
	DWT-Net	29.664	0.877	32.227	0.924	33.738	0.943	34.787	0.953	35.652	0.960	36.511	0.966	37.653	0.973	39.287	0.980
	DSPW-Net$_{NS}$	30.255	0.889	32.829	0.932	34.310	0.949	35.300	0.958	36.105	0.964	36.876	0.970	37.865	0.975	39.213	0.981
	DSPW-Net$_{DS}$	30.401	0.890	33.082	0.933	34.646	0.950	35.716	0.959	36.594	0.965	37.451	0.970	38.589	0.976	40.196	0.982
	DSPW-Net$_{OS}$	30.436	0.890	33.115	0.933	34.680	0.950	35.748	0.959	36.627	0.965	37.483	0.971	38.622	0.976	40.223	0.982
	DSPW-Net	30.526	0.892	33.204	0.934	34.774	0.951	35.847	0.960	36.723	0.966	37.583	0.971	38.717	0.976	40.316	0.982

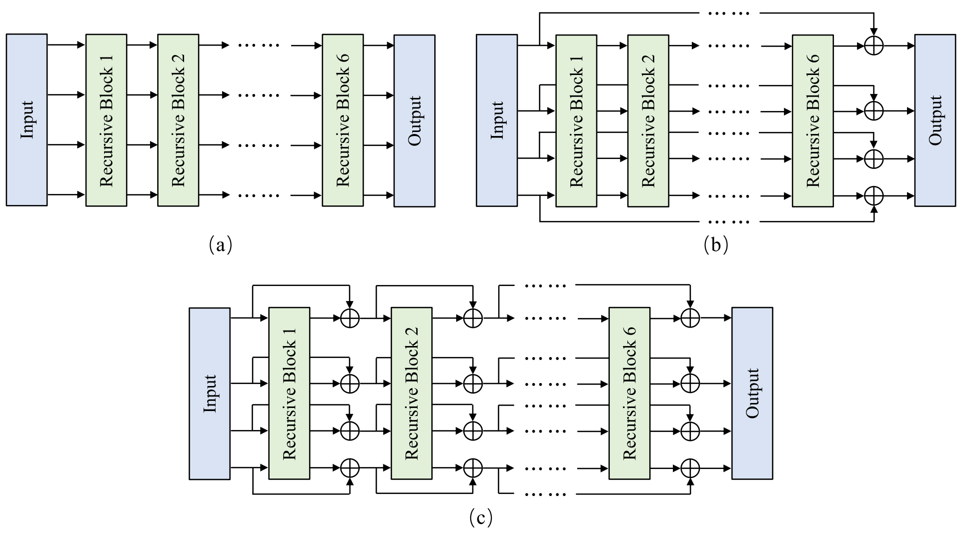

Fig. 11. The additional three different residual learning methods explored in this work: (a) no skip connection; (b) one skip connection; and (c) distinct-source skip connection.

compressed images. To this end, 23 pristine images in the CIDIQ dataset [89] were first uploaded to Facebook[3] and then downloaded to the local drive. Note that this is the most common way in our daily lives to share images/photos. Since social media networks such as Facebook mostly re-compress photos when uploading, these downloaded images are actually the compressed images that contain ringing/blocking artifacts though very minor. Accordingly, we use as testing data the 23 downloaded images and compare DSPW-Net against five deblocking methods which also consist of single unified models: SA-DCT, DnCNN, AGARNet, QCN, and MDU. The results in terms of average PSNR and SSIM values are presented in Table 8, and a visual comparison of different algorithm outputs is shown in Fig. 12. Also included in Fig. 12 are the downloaded compressed image and the ground truth image for reference.

Again, we observe that DSPW-Net demonstrates better results than other deblocking methods. On this test, it seems that AGARNet performs less effectively on these downloaded images according to Table 8, perhaps suggesting that AGARNet has not been well trained to take into account high-QF images. This finding is also in accord with the results in Fig. 9, in which the performance of AGARNet drops considerably when QF is greater than 80. Although we are able to show only a limited set of demonstrative images, overall, DSPW-Net shows either highly competitive or superior deblocking performance as compared to existing methods.

[3] https://www.facebook.com

4. Conclusion

In this paper, we presented a unified model for JPEG compression artifact reduction based on introducing deep convolutional neural networks to the complex version of steerable pyramid wavelet transform. Our model separately restores the Y channel image and the Cb/Cr channel image via two restoration branches. The Y-channel restoration branch contains six recursive blocks and takes as input the multiple-scale, multiple-orientation wavelet coefficients to predict the uncompressed wavelet subbands. The CbCr-channel restoration branch

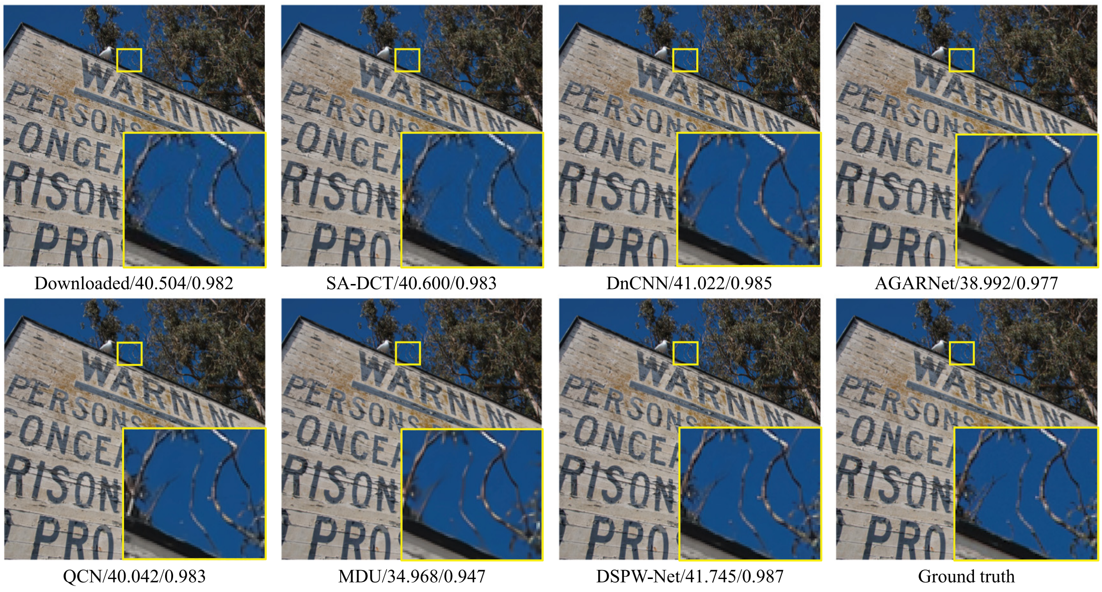

Fig. 12. Visual comparison of various deblocking methods applied on a pristine image *final03* in the CIDIQ dataset [89]. Color was added to the DnCNN, AGARNet, and MDU output by using the restored Cb/Cr channels generated by DSPW-Net. The image was first uploaded and then downloaded from the Facebook website to generate the compressed version. The corresponding PSNR and SSIM values are presented at the bottom of each image. Notice that DSPW-Net performs particularly well at recreating the tree branches as compared to the other methods.

Table 7
Performance of DSPW-Net trained using different network/training parameters.

Dataset	Method	QF=10		QF=20		QF=30		QF=40		QF=50		QF=60		QF=70		QF=80	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
LIVE	$S = 1$	30.058	0.838	32.425	0.897	33.828	0.922	34.819	0.936	35.645	0.945	36.474	0.953	37.610	0.962	39.265	0.972
	$S = 2$	30.107	0.839	32.471	0.898	33.877	0.923	34.867	0.936	35.692	0.945	36.521	0.953	37.657	0.962	39.311	0.972
	$R = 2$	30.080	0.839	32.450	0.898	33.856	0.922	34.849	0.936	35.676	0.945	36.507	0.953	37.641	0.962	39.296	0.972
	$R = 4$	30.136	0.840	32.504	0.899	33.913	0.923	34.903	0.936	35.729	0.946	36.561	0.953	37.695	0.962	39.344	0.972
	$R = 8$	30.159	0.840	32.523	0.899	33.930	0.923	34.919	0.937	35.744	0.946	36.574	0.954	37.707	0.962	39.359	0.972
	$\lambda = 0.01$	30.143	0.842	32.509	0.900	33.914	0.924	34.905	0.937	35.730	0.946	36.561	0.954	37.696	0.962	39.344	0.972
	$\lambda = 0.1$	30.046	0.844	32.422	0.901	33.842	0.925	34.843	0.937	35.676	0.946	36.513	0.954	37.655	0.963	39.318	0.972
	$\lambda = 1$	29.846	0.845	32.282	0.901	33.742	0.925	34.765	0.938	35.611	0.947	36.457	0.954	37.610	0.963	39.284	0.972
	DSPW-Net	30.155	0.840	32.518	0.899	33.927	0.923	34.918	0.937	35.743	0.946	36.574	0.954	37.707	0.962	39.360	0.972
BSD100	$S = 1$	29.550	0.808	31.740	0.875	33.047	0.904	33.980	0.920	34.800	0.933	35.627	0.943	36.783	0.955	38.502	0.969
	$S = 2$	29.588	0.809	31.775	0.875	33.082	0.904	34.014	0.921	34.833	0.933	35.661	0.944	36.816	0.956	38.536	0.969
	$R = 2$	29.568	0.808	31.757	0.875	33.066	0.904	33.999	0.921	34.819	0.933	35.649	0.944	36.806	0.956	38.524	0.969
	$R = 4$	29.610	0.809	31.800	0.876	33.105	0.905	34.037	0.921	34.856	0.933	35.685	0.944	36.843	0.956	38.545	0.969
	$R = 8$	29.625	0.809	31.812	0.876	33.116	0.905	34.045	0.921	34.865	0.934	35.692	0.944	36.847	0.956	38.563	0.969
	$\lambda = 0.01$	29.613	0.811	31.798	0.877	33.104	0.906	34.034	0.922	34.854	0.934	35.682	0.944	36.837	0.956	38.540	0.969
	$\lambda = 0.1$	29.507	0.815	31.699	0.879	33.024	0.907	33.965	0.923	34.788	0.935	35.622	0.945	36.785	0.956	38.506	0.969
	$\lambda = 1$	29.283	0.816	31.532	0.880	32.902	0.907	33.869	0.923	34.706	0.935	35.552	0.945	36.727	0.956	38.460	0.969
	DSPW-Net	29.622	0.809	31.809	0.876	33.115	0.905	34.044	0.921	34.865	0.933	35.693	0.944	36.849	0.956	38.567	0.969
CSIQ	$S = 1$	30.369	0.858	32.883	0.910	34.344	0.931	35.344	0.942	36.156	0.950	36.972	0.957	38.072	0.965	39.646	0.973
	$S = 2$	30.421	0.859	32.929	0.910	34.385	0.931	35.387	0.942	36.195	0.950	37.012	0.957	38.115	0.965	39.695	0.973
	$R = 2$	30.401	0.858	32.914	0.910	34.372	0.931	35.374	0.942	36.182	0.950	37.000	0.957	38.102	0.965	39.674	0.973
	$R = 4$	30.449	0.859	32.959	0.911	34.421	0.931	35.420	0.943	36.231	0.951	37.047	0.957	38.148	0.965	39.721	0.974
	$R = 8$	30.472	0.860	32.977	0.911	34.436	0.931	35.437	0.943	36.247	0.951	37.061	0.957	38.162	0.965	39.723	0.974
	$\lambda = 0.01$	30.458	0.861	32.969	0.911	34.430	0.932	35.431	0.943	36.243	0.951	37.058	0.958	38.159	0.965	39.721	0.974
	$\lambda = 0.1$	30.372	0.863	32.905	0.912	34.378	0.932	35.388	0.944	36.205	0.951	37.024	0.958	38.129	0.965	39.703	0.974
	$\lambda = 1$	30.183	0.864	32.785	0.913	34.290	0.933	35.320	0.944	36.146	0.951	36.973	0.958	38.086	0.965	39.667	0.974
	DSPW-Net	30.468	0.860	32.974	0.911	34.435	0.931	35.435	0.943	36.247	0.951	37.061	0.957	38.162	0.965	39.736	0.974
Urban100	$S = 1$	30.261	0.887	32.947	0.931	34.527	0.949	35.605	0.958	36.482	0.965	37.342	0.970	38.474	0.976	40.073	0.982
	$S = 2$	30.423	0.890	33.093	0.933	34.668	0.950	35.745	0.959	36.624	0.965	37.481	0.970	38.613	0.976	40.217	0.982
	$R = 2$	30.271	0.887	32.949	0.931	34.536	0.949	35.621	0.958	36.507	0.965	37.374	0.970	38.520	0.976	40.129	0.982
	$R = 4$	30.495	0.891	33.173	0.934	34.746	0.950	35.823	0.960	36.702	0.966	37.563	0.971	38.698	0.976	40.289	0.982
	$R = 8$	30.563	0.892	33.236	0.934	34.799	0.951	35.864	0.960	36.736	0.966	37.591	0.971	38.724	0.976	40.324	0.982
	$\lambda = 0.01$	30.492	0.892	33.164	0.934	34.728	0.951	35.794	0.960	36.666	0.966	37.525	0.971	38.658	0.976	40.249	0.982
	$\lambda = 0.1$	30.432	0.894	33.109	0.935	34.680	0.951	35.750	0.960	36.626	0.966	37.488	0.971	38.624	0.977	40.223	0.982
	$\lambda = 1$	30.306	0.895	33.014	0.935	34.604	0.952	35.685	0.960	36.570	0.966	37.437	0.971	38.580	0.977	40.185	0.982
	DSPW-Net	30.526	0.892	33.204	0.934	34.774	0.951	35.847	0.960	36.723	0.966	37.583	0.971	38.717	0.976	40.316	0.982

Table 8
Average PSNR and SSIM values of DSPW-Net vs. competing methods tested on real-world compressed images downloaded from Facebook.

	Downloaded	SA-DCT [23]	DnCNN [48]	AGARNet [71]	QCN [53]	MDU [70]	DSPW-Net
PSNR	44.468	44.750	45.208	44.431	45.085	40.778	45.747
SSIM	0.986	0.987	0.988	0.986	0.988	0.974	0.989

contains six recursive U-Net architectures, and takes as input the compressed Cb/Cr channel as well as the restored Y-channel to produce the restored Cb/Cr channel. To enable the two branches to work effectively without requiring the prior knowledge about the encoding parameters, the QF-related features are incorporated to allow the network to learn complex mappings between the pristine and distorted images. By using skip connections in each branch to train deeper models and by adopting recursive blocks to share parameters, DSPW-Net has fewer parameters while still achieving state-of-the-art restoration performance. We presented a comprehensive evaluation on various network design choices and believe that the thorough analysis can benefit other researchers in this field. However, despite the effectiveness of the proposed DSPW-Net in reducing JPEG compression artifact in natural-scene images, whether or not it can be applied to other types of distortions (e.g., JPEG2000 compression, noise, and blur, etc.) or other types of images (e.g., medical image, remote sensing image, etc.) remains an open question, and we will take into account these issues in our future work.

CRediT authorship contribution statement

Yi Zhang: Conceptualization, Methodology, Software, Investigation, Writing – original draft. **Damon M. Chandler:** Software, Data curation, Visualization, Formal analysis, Writing – review & editing. **Xuanqin Mou:** Validation, Resources, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 61901355, 62071375, 62271384) and the China Postdoctoral Science Foundation (Grant No. 2018M640991).

References

[1] G.K. Wallace, The JPEG still picture compression standard, IEEE Trans. Consum. Electron. 38 (1) (1992) xviii–xxxiv.
[2] G.J. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, Overview of the high efficiency video coding (HEVC) standard, IEEE Trans. Circuits Syst. Video Technol. 22 (12) (2012) 1649–1668.
[3] T. Wiegand, G.J. Sullivan, G. Bjontegaard, A. Luthra, Overview of the H. 264/AVC video coding standard, IEEE Trans. Circuits Syst. Video Technol. 13 (7) (2003) 560–576.
[4] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G.J. Sullivan, J.-R. Ohm, Overview of the versatile video coding (VVC) standard and its applications, IEEE Trans. Circuits Syst. Video Technol. 31 (10) (2021) 3736–3764.
[5] Y.-W. Lee, J.-H. Kim, Y.-J. Choi, B.-G. Kim, CNN-based approach for visual quality improvement on HEVC, in: 2018 IEEE International Conference on Consumer Electronics, ICCE, IEEE, 2018, pp. 1–3.
[6] J.-H. Lee, Y.-W. Lee, D. Jun, B.-G. Kim, Efficient color artifact removal algorithm based on high-efficiency video coding (HEVC) for high-dynamic range video sequences, IEEE Access 8 (2020) 64099–64111.
[7] S. Kuanar, K. Rao, C. Conly, N. Gorey, Deep learning based HEVC in-loop filter and noise reduction, Signal Process., Image Commun. 99 (2021) 116409.
[8] S. Bouaafia, S. Messaoud, R. Khemiri, F.E. Sayadi, VVC in-loop filtering based on deep convolutional neural network, Comput. Intell. Neurosci. 2021 (2021).
[9] C.D. Pham, C. Fu, J. Zhou, Deep learning based spatial-temporal in-loop filtering for versatile video coding, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 1861–1865.
[10] Z. Wang, D. Liu, S. Chang, Q. Ling, Y. Yang, T.S. Huang, D3: Deep dual-domain based fast restoration of JPEG-compressed images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2764–2772.
[11] T.P. O'Rourke, R.L. Stevenson, Improved image decompression for reduced transform coding artifacts, IEEE Trans. Circuits Syst. Video Technol. 5 (6) (1995) 490–499.
[12] T. Meier, K. Ngan, G. Crebbin, Reduction of blocking artifacts in image and video coding, IEEE Trans. Circuits Syst. Video Technol. 9 (3) (1999) 490–500.
[13] L. Ma, D. Zhao, W. Gao, Learning-based image restoration for compressed images, Signal process.: Image commun. 27 (1) (2012) 54–65.
[14] X. Zhang, R. Xiong, X. Fan, S. Ma, W. Gao, Compression artifact reduction by overlapped-block transform coefficient estimation with block similarity, IEEE Trans. Image Process. 22 (12) (2013) 4613–4626.
[15] S.D. Kim, J. Yi, H.M. Kim, J.B. Ra, A deblocking filter with two separate modes in block-based video coding, IEEE Trans. Circuits Syst. Video Technol. 9 (1) (1999) 156–160.
[16] H.W. Park, Y.L. Lee, A postprocessing method for reducing quantization effects in low bit-rate moving picture coding, IEEE Trans. Circuits Syst. Video Technol. 9 (1) (1999) 161–171.
[17] G. Zhai, W. Zhang, X. Yang, W. Lin, Y. Xu, Efficient image deblocking based on postfiltering in shifted windows, IEEE Trans. Circuits Syst. Video Technol. 18 (1) (2008) 122–126.
[18] B. Ramamurthi, A. Gersho, Nonlinear space-variant postprocessing of block coded images, IEEE Trans. Acoust. Speech Signal Process. 34 (5) (1986) 1258–1268.
[19] C. Wang, J. Zhou, S. Liu, Adaptive non-local means filter for image deblocking, Signal Process., Image Commun. 28 (5) (2013) 522–530.
[20] N.C. Francisco, N.M. Rodrigues, E.A. Da Silva, S.M. De Faria, A generic post-deblocking filter for block based image compression algorithms, Signal Process., Image Commun. 27 (9) (2012) 985–997.
[21] S. Minami, A. Zakhor, An optimization approach for removing blocking effects in transform coding, IEEE Trans. Circuits Syst. Video Technol. 5 (2) (1995) 74–82.
[22] Y. Luo, R.K. Ward, Removing the blocking artifacts of block-based DCT compressed images, IEEE Trans. Image Process. 12 (7) (2003) 838–842.
[23] A. Foi, V. Katkovnik, K. Egiazarian, Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images, IEEE Trans. Image Pocess. 16 (5) (2007) 1395–1411.
[24] G.A. Triantafyllidis, D. Tzovaras, M.G. Strintzis, Blocking artifact detection and reduction in compressed data, IEEE Trans. Circuits Syst. Video Technol. 12 (10) (2002) 877–890.
[25] A. Nosratinia, Enhancement of JPEG-compressed images by re-application of JPEG, J. VLSI Signal Process. Syst. signal image video Technol. 27 (1–2) (2001) 69–79.
[26] R. Samadani, A. Sundararajan, A. Said, Deringing and deblocking DCT compression artifacts with efficient shifted transforms, in: 2004 International Conference on Image Processing, Vol. 3, 2004. ICIP'04, IEEE, 2004, pp. 1799–1802.
[27] D. Sun, W.-K. Cham, Postprocessing of low bit-rate block DCT coded images based on a fields of experts prior, IEEE Trans. Image Process. 16 (11) (2007) 2743–2751.
[28] T. Li, X. He, L. Qing, Q. Teng, H. Chen, An iterative framework of cascaded deblocking and superresolution for compressed images, IEEE Trans. Multimed. 20 (6) (2017) 1305–1320.
[29] J. Ren, J. Liu, M. Li, W. Bai, Z. Guo, Image blocking artifacts reduction via patch clustering and low-rank minimization, in: 2013 Data Compression Conference, IEEE, 2013, p. 516.
[30] M. Yin, J. Gao, Y. Sun, S. Cai, Blocky artifact removal with low-rank matrix recovery, in: IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2014, pp. 1996–2000.
[31] Y. Yang, N.P. Galatsanos, Removal of compression artifacts using projections onto convex sets and line process modeling, IEEE Trans. Image Process. 6 (10) (1997) 1345–1357.
[32] X. Zhang, W. Lin, R. Xiong, X. Liu, S. Ma, W. Gao, Low-rank decomposition-based restoration of compressed images via adaptive noise estimation, IEEE Trans. Image Process. 25 (9) (2016) 4158–4171.
[33] X. Zhang, R. Xiong, S. Ma, W. Gao, Reducing blocking artifacts in compressed images via transform-domain non-local coefficients estimation, in: 2012 IEEE International Conference on Multimedia and Expo, IEEE, 2012, pp. 836–841.
[34] C. Jung, L. Jiao, H. Qi, T. Sun, Image deblocking via sparse representation, Signal Process., Image Commun. 27 (6) (2012) 663–677.
[35] H. Chang, M.K. Ng, T. Zeng, Reducing artifacts in JPEG decompression via a learned dictionary, IEEE Trans. Signal Process. 62 (3) (2013) 718–728.
[36] C.-H. Yeh, L.-W. Kang, Y.-W. Chiou, C.-W. Lin, S.-J.F. Jiang, Self-learning-based post-processing for image/video deblocking via sparse representation, J. Vis. Commun. Image Represent. 25 (5) (2014) 891–903.
[37] J. Mu, X. Zhang, R. Xiong, S. Ma, W. Gao, Adaptive multi-dimension sparsity based coefficient estimation for compression artifact reduction, in: 2016 IEEE International Conference on Multimedia and Expo, IEEE, 2016, pp. 1–6.
[38] L. Wang, X. Zhou, C. Wang, B. Jiang, Post-processing for JPEG-coded image deblocking via sparse representation and adaptive residual threshold, KSII Trans. Internet Inform. Syst. 11 (3) (2017).
[39] X. Liu, X. Wu, J. Zhou, D. Zhao, Data-driven sparsity-based restoration of JPEG-compressed images in dual transform-pixel domain, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5171–5178.
[40] X. Liu, X. Wu, J. Zhou, D. Zhao, Data-driven soft decoding of compressed images in dual transform-pixel domain, IEEE Trans. Image Process. 25 (4) (2016) 1649–1659.
[41] J. Zhang, S. Ma, Y. Zhang, W. Gao, Image deblocking using group-based sparse representation and quantization constraint prior, in: 2015 IEEE International Conference on Image Processing, IEEE, 2015, pp. 306–310.
[42] C. Zhao, J. Zhang, S. Ma, X. Fan, Y. Zhang, W. Gao, Reducing image compression artifacts by structural sparse representation and quantization constraint prior, IEEE Trans. Circuits Syst. Video Technol. 27 (10) (2016) 2057–2071.

[43] J. Zhang, R. Xiong, C. Zhao, Y. Zhang, S. Ma, W. Gao, CONCOLOR: Constrained non-convex low-rank model for image deblocking, IEEE Trans. Image Process. 25 (3) (2016) 1246–1259.
[44] X. Liu, G. Cheung, X. Wu, D. Zhao, Random walk graph Laplacian-based smoothness prior for soft decoding of JPEG images, IEEE Trans. Image Process. 26 (2) (2016) 509–524.
[45] H. Chen, X. He, L. Qing, S. Xiong, T.Q. Nguyen, DPW-SDNet: Dual pixel-wavelet domain deep CNNs for soft decoding of JPEG-compressed images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018.
[46] C. Dong, Y. Deng, C. Change Loy, X. Tang, Compression artifacts reduction by a deep convolutional network, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 576–584.
[47] Y. Chen, T. Pock, Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration, IEEE Trans. Pattern Anal. Mach. Intell. 39 (6) (2016) 1256–1272.
[48] K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising, IEEE Trans. Image Process. 26 (7) (2017) 3142–3155.
[49] L. Cavigelli, P. Hager, L. Benini, CAS-CNN: A deep convolutional neural network for image compression artifact suppression, in: 2017 International Joint Conference on Neural Networks, IEEE, 2017, pp. 752–759.
[50] Y. Tai, J. Yang, X. Liu, C. Xu, MemNet: A persistent memory network for image restoration, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4539–4547.
[51] B. Zheng, R. Sun, X. Tian, Y. Chen, S-Net: A scalable convolutional neural network for JPEG compression artifact reduction, J. Electron. Imaging 27 (4) (2018) 043037.
[52] Z. Jin, M.Z. Iqbal, W. Zou, X. Li, E. Steinbach, Dual-stream multi-path recursive residual network for JPEG image compression artifacts reduction, IEEE Trans. Circuits Syst. Video Technol. 31 (2) (2020) 467–479.
[53] J. Li, Y. Wang, H. Xie, K.-K. Ma, Learning a single model with a wide range of quality factors for JPEG image artifacts removal, IEEE Trans. Image Process. 29 (2020) 8842–8854.
[54] H. Chen, X. He, H. Yang, L. Qing, Q. Teng, A feature-enriched deep convolutional neural network for JPEG image compression artifacts reduction and its applications, IEEE Trans. Neural Netw. Learn. Syst. 33 (1) (2021) 430–444.
[55] L. Ma, P. Peng, P. Xing, Y. Wang, Y. Tian, Reducing image compression artifacts for deep neural networks, in: 2021 Data Compression Conference, DCC, IEEE, 2021, p. 355.
[56] C. Mou, J. Zhang, Z. Wu, Dynamic attentive graph learning for image restoration, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 4328–4337.
[57] J. Guo, H. Chao, One-to-many network for visually pleasing compression artifacts reduction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3038–3047.
[58] X. Fu, Z. Zha, F. Wu, X. Ding, J. Paisley, JPEG artifacts reduction via deep convolutional sparse coding, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 2501–2510.
[59] I. Goodfellow, J.P. Abadie, M. Mirza, B. Xu, D.W. Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[60] L. Galteri, L. Seidenari, M. Bertini, A.D. Bimbo, Deep generative adversarial compression artifact removal, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4826–4835.
[61] L. Galteri, L. Seidenari, M. Bertini, A.D. Bimbo, Deep universal generative adversarial compression artifact removal, IEEE Trans. Multimed. 21 (8) (2019) 2131–2145.
[62] Z. Zhao, Q. Sun, H. Yang, H. Qiao, Z. Wang, D.O. Wu, Compression artifacts reduction by improved generative adversarial networks, EURASIP J. Image Video Process. 2019 (1) (2019) 1–7.
[63] B. Zheng, Y. Chen, X. Tian, F. Zhou, X. Liu, Implicit dual-domain convolutional network for robust color image compression artifact reduction, IEEE Trans. Circuits Syst. Video Technol. 30 (11) (2019) 3982–3994.
[64] X. Zhang, W. Yang, Y. Hu, J. Liu, DMCNN: Dual-domain multi-scale convolutional neural network for compression artifacts removal, in: 2018 25th IEEE International Conference on Image Processing, IEEE, 2018, pp. 390–394.
[65] J. Guo, H. Chao, Building dual-domain representations for compression artifacts reduction, in: European Conference on Computer Vision, Springer, 2016, pp. 628–644.
[66] M. Ehrlich, L. Davis, S.-N. Lim, A. Shrivastava, Quantization guided JPEG artifact correction, in: European Conference on Computer Vision, Springer, 2020, pp. 293–309.
[67] P. Liu, H. Zhang, K. Zhang, L. Lin, 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.
[68] S. Zini, S. Bianco, R. Schettini, Deep residual autoencoder for blind universal JPEG restoration, IEEE Access 8 (2020) 63283–63294.
[69] X. Fu, X. Wang, A. Liu, J. Han, Z. Zha, Learning dual priors for JPEG compression artifacts removal, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 4086–4095.
[70] X. Fu, M. Wang, X. Cao, X. Ding, Z. Zha, A model-driven deep unfolding method for JPEG artifacts removal, IEEE Trans. Neural Netw. Learn. Syst. 33 (11) (2021) 6802–6816.
[71] Y. Kim, J.W. Soh, N.I. Cho, AGARNet: Adaptively gated JPEG compression artifacts removal network for a wide range quality factor, IEEE Access 8 (2020) 20160–20170.
[72] J. Jiang, K. Zhang, R. Timofte, Towards flexible blind JPEG artifacts removal, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 4997–5006.
[73] H. Sheikh, M. Sabir, A. Bovik, A statistical evaluation of recent full reference image quality assessment algorithms, vol. 15, no. 11, 2006, pp. 1349–1364.
[74] J.-B. Huang, A. Singh, N. Ahuja, Single image super-resolution from transformed self-exemplars, 2015, http://dx.doi.org/10.1109/CVPR.2015.7299156.
[75] E.C. Larson, D.M. Chandler, Most apparent distortion: Full-reference image quality assessment and the role of strategy, J. Electron. Imaging 19 (1) (2010) 011006.
[76] O. Ronneberger, P. Fischer, 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.
[77] J. Portilla, E.P. Simoncelli, A parametric texture model based on joint statistics of complex wavelet coefficients, Int. J. Comput. Vis. 40 (1) (2000) 49–70.
[78] B.T. Series, Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios, International Telecommunication Union, Radiocommunication Sector, 2011.
[79] M. Everingham, L.V. Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, Int. J. Comput. Vis. 88 (2) (2010) 303–338.
[80] W.T. Freeman, E.H. Adelson, The design and use of steerable filters, IEEE Trans. Pattern Anal. Mach. Intell. 13 (9) (1991) 891–906.
[81] E.P. Simoncelli, J. Portilla, Texture characterization via joint statistics of wavelet coefficient magnitudes, in: Proceedings., International Conference on Image Processing, Vol. 1, IEEE, 1998, pp. 62–66.
[82] E.P. Simoncelli, W.T. Freeman, The steerable pyramid: A flexible architecture for multi-scale derivative computation, in: Proceedings., International Conference on Image Processing, Vol. 3, IEEE, 1995, pp. 444–447.
[83] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in: IEEE International Conference on Computer Vision, Vol. 2, ICCV, 2001, pp. 416–423.
[84] E. Agustsson, 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.
[85] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, L. Zhang, Waterloo exploration database: New challenges for image quality assessment models, IEEE Trans. Image Process. 26 (2) (2016) 1004–1016.
[86] Z. Wang, A.C. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment: From error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
[87] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[88] C. Silvia, G. Francesca, S. Raimondo, No reference image quality classification for JPEG-distorted images, Digit. Signal Process. 30 (2014) 86–100.
[89] X. Liu, M. Pedersen, J.Y. Hardeberg, CID:IQ–a new image quality database, in: International Conference on Image and Signal Processing, Springer, 2014, pp. 193–202.
[90] R. Zhang, P. Isola, A.A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 586–595.
[91] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90.
[92] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.
[93] Y. Tai, J. Yang, X. Liu, Image super-resolution via deep recursive residual network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3147–3155.
[94] W.-S. Lai, J.-B. Huang, N. Ahuja, M.-H. Yang, Fast and accurate image super-resolution with deep laplacian pyramid networks, IEEE Trans. Pattern Anal. Mach. Intell. 41 (11) (2018) 2599–2613.