Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

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

Full length article

Multi-domain residual encoder–decoder networks for generalized 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:
Artifact reduction
Residual network
Compression artifact
Quality factor estimation

ABSTRACT

A fundamental requirement for designing compression artifact reduction techniques is to restore the artifact free image from its compressed version regardless of the compression level. Most existing algorithms require the prior knowledge of JPEG encoding parameters to operate effectively. Although there are works that attempt to train universal models to deal with different compression levels, some JPEG quality factors (QF) are still missing. To overcome these potential limitations, in this paper, we present a generalized JPEG-compression artifact reduction framework that relies on improved QF estimator and rectified networks to take into account all possible QF values. Our method, called a generalized compression artifact reducer (G-CAR), first predicts QF by analyzing luminance patches with high activity. Then, based on the estimated QF, images are adaptively restored by the cascaded residual encoder–decoder networks learned in multiple domains. Results tested on six benchmark datasets demonstrate the effectiveness of our proposed model.

1. Introduction

With the rapid development of modern cameras and digital imaging technology, billions of images/videos are captured, stored, and shared on the Internet every day. These images/videos have to be compressed for transmission and storage in order to save both bandwidth and in-device resources. Apart from a few cases where lossless compression is adopted (e.g., medical imaging and technical drawing), lossy compression, such as JPEG [1] and JPEG2000 [2], has been widely used (e.g., online education, entertainment video streaming, wireless surveillance, remote conference, etc.) to achieve a much higher compression ratio. These coding algorithms generally work by quantizing and encoding images in the transform domain, giving rising to code streams that mostly contain zero-value coefficients, and hence encode images via limited amount of data. For example, the discrete cosine transform (DCT) is used in JPEG while the discrete wavelet transform (DWT) is used in JPEG2000. Quantization of these DCT or DWT coefficients produces inaccurate superpositions of the respective basis functions, which ultimately manifest as undesired image artifacts such as blockiness, ringing, and blurring that are especially visible at low-bit rates. JPEG2000 typically suffers from blurring due to the loss of high-frequency components whereas JPEG suffers from blocking due to the individual treatment of adjacent coding blocks. These compression artifacts are not only visually unpleasant, but also have a negative impact on various image processing and computer vision algorithms that take compressed images as input. Thus, algorithms which can effectively reduce the amount or visibility of compression artifacts in images/videos are in great demand.

In this paper, we focus on reducing compression artifacts in JPEG compressed images. Although a large number of image restoration algorithms have been reported to successfully remove the different kinds of noise and blur artifacts in images (see [3,4] for a survey), designing restoration techniques to combat JPEG compression artifacts still remains quite challenging. This is due to the fact that the non-linearity of quantization makes the resulting noise non-stationary and signal dependent [5]. For example, after quantization, *banding effects* become visible in smooth regions and ringing artifacts appear around sharp edges. Moreover, in comparing with JPEG2000 that typically introduces ringing and blurring artifacts to images, JPEG additionally introduces a very distinct compression artifact, the blocking artifact, which occurs due to the independent process of non-overlapping 8×8 image blocks and consequently the boundaries between coding blocks become discontinuous. Thus, restoration algorithms (e.g., [6–9]) that model quantization noises as signal independent often perform less effectively on compressed images.

To overcome the blocking artifact problem, a number of deblocking/soft-decoding approaches have been proposed. Among

☆ This paper has been recommended for acceptance by Zicheng Liu.
* Corresponding author.
E-mail address: yi.zhang.osu@xjtu.edu.cn (Y. Zhang).

https://doi.org/10.1016/j.jvcir.2021.103425
Received 13 April 2021; Received in revised form 28 August 2021; Accepted 22 December 2021
Available online 19 January 2022

these approaches, one idea is to perform filtering operations along boundaries in the spatial and/or transform domain. Typical spatial-domain filtering algorithms (e.g., [10–12]) select a filtering mode according to the pixel behavior of the block boundary and then apply one-dimensional (1-D) horizontal and vertical filtering or two-dimensional (2-D) spatially adaptive filtering across different image regions. Other spatial-domain algorithms include employing postfiltering in shifted windows of image blocks [13], non-linear space-variant filtering [14], adaptive nonlocal means filtering [15], adaptive bilateral filtering [16], etc.

In comparison, transform-domain filtering approaches directly adjust DCT coefficients. Such algorithms include that considers the correlation between boundary pixel values of two neighboring blocks [17,18], the masking effect in the human visual system (HVS) [19], the local ac coefficient regularization of shifted blocks [13], the local visibility measure of blocking artifacts at block edges [20], and the optimal correction of the DCT coefficients [21], etc. As the other approach, the JPEG process itself is employed to reduce the compression artifacts by reapplying JPEG compression to the various shifted versions of the compressed image and then average the reapplied results [22,23].

Another type of the deblocking/soft-decoding approach treats compression artifact reduction as an ill-posed inverse problem, where the prior knowledge about high-quality images, compression algorithms, and compression parameters is employed to guide the restoration process [24]. Typical image priors include: the field of experts prior [25], the low-rank prior [26–28], the quantization constraint prior [29,30], non-local similarity [9,31], the sparse representation prior [32–36], etc. Moreover, some approaches employ more than one image prior to restore compressed images. For example, the sparse representation and quantization constraint priors were used in [37–40]; the low-rank and quantization constraint priors were used in [41]; the Laplacian prior, sparsity prior, and graph-signal smoothness prior were used in [42]. However, as mentioned in [24], most of the image-prior-based deblocking algorithms are time-consuming due to the complex optimization process.

With the wide/spread utilization of deep learning techniques in recent years, a third type of the deblocking/soft-decoding approach has appeared, which relies on a deep convolutional neural network (CNN) as well as some traditional image transforms, priors, and constraints. After the first CNN-based deblocking algorithm (ARCNN [43]) was proposed, successive works have mainly focused on two important issues: better network architectures and better utilization of the transform domain information. For example, the feed-forward denoising CNN (DnCNN) was proposed in [44]; the residual encoder–decoder network (RED-Net) was proposed in [45]; the compression artifact suppression CNN (CAS-CNN) was proposed in [46]; the one-to-many network was proposed in [47]; the trainable nonlinear reaction diffusion (TNRD) model was proposed in [48]; the persistent memory network (MemNet) was proposed in [49]; the scalable CNN (S-Net) was proposed in [50]; the deep convolutional sparse coding (DCSC) network was proposed in [51]; the deep residual auto-encoder was proposed in [52]; the generative adversarial networks (GANs) [53] were employed in [54–56], and so forth. As JPEG compression is not optimal, image transforms such as DCT and DWT were also embedded in the network design to further explore and utilize the redundant information neglected by the JPEG encoder. Consequently, some dual-domain deblocking models such as D^3 [5], DMCNN [57], DDCN [58], MWCNN [59], and DPW-SDNet [24] have been proposed. Meanwhile, to generate visually-comfortable restoration results, various loss functions such as adversarial loss [60,61], structure similarity (SSIM [62]) loss, perceptual loss [63,64], edge emphasized loss [65], and JPEG loss [47] were employed for network training.

Despite the promising results achieved thus far, one important issue neglected by most existing approaches is the fact that in most cases the encoding parameter of a compressed image is unknown. Since most deblocking algorithms require this information to operate effectively, their practical applications are restricted. To the best of our knowledge, among the CNN-based deblocking approaches, only a few works attempt to address this issue. One is the DnCNN model [44] which was trained on compressed images with quality factors (QFs) ranging from 5 to 99. The other one is the multiple GANs model [55] which operates by incorporating a QF estimator such that different GANs are applied based on different estimated QF values. Though effective, both models suffer from potential limitations. For example, since DnCNN was trained on compressed images with a wide range of QF values, larger restoration errors might occur as compared with training the same model on images with a single fixed QF value. In comparison, although in [55] each generator of GANs was trained on images with fixed QF values, the algorithm performance is influenced by the accuracy of the QF estimator. Also, the compression artifacts of images corresponding to other QF values not observed in the training data may not be effectively removed, especially when QF value is small (i.e., images are heavily compressed), because the algorithm always employs a generative model whose trained QF is closer to the output of the QF estimator to perform the deblocking task. Finally, both models operate in the pixel domain only, while many existing works (e.g., [24,38,58], etc.) suggest that incorporating analyses in other domains (e.g., DCT or DWT) can possibly benefit the restoration performance.

Based on the abovementioned points, in this paper we propose a generalized JPEG compression artifact reduction framework based on learning a cascaded residual encoder–decoder network (CRED-Net) in the pixel, DCT, and DWT domains. Our method, called a generalized compression artifact reducer (G-CAR), operates via two main stages: (1) JPEG-compression quality factor estimation; and (2) QF-specific compression artifact reduction, as shown in Fig. 1. In the first stage, the luminance channel (Y channel of the YCbCr space) of an image is divided into non-overlapping patches, and patches with high activity are selected for QF estimation. The QF of the whole image is then computed as the rounded average of all estimated QF values. In the second stage, corresponding multi-domain CRED-Nets are applied to the Y channel and corresponding CbCr-Nets are applied to the Cb and Cr channels to perform the compression artifact reduction task. Specifically, for Y channel restoration, if the estimated QF equals one of the predefined values or fall into one of the predefined ranges (please refer to Section 2.2.3 for more details), then a single corresponding multi-domain CRED-Net is directly applied to obtain the deblocking result. Otherwise, the multi-domain CRED-Net is first applied, and then its output along with the original input are fed into another CRED-Net for further restoration. These additional CRED-Nets (also called rectified networks) are trained on images with small QF variations (e.g., QF $\in [8, 9], [11, 14], etc.$) to overcome the potential limitation that a single network cannot effectively address images with all compression levels. For Cb and Cr channels restoration, the optimal CbCr-Net is applied based on investigating the specific range into which the estimated QF value falls. Finally, the restored Y, Cb, and Cr channels are converted back to the RGB space to produce the restored image.

Compared with existing deblocking approaches, our method has several appealing properties. First, it is a blind and generalized framework, meaning that the algorithm requires no prior knowledge of the encoding parameters. Like [55], we also trained a QF estimator to indicate compression quality. However, different from [55], our method on the one hand focuses on quality-relevant regions (instead of all image patches) for more accurate QF estimation. On the other hand, we additionally train rectified networks to take into account all compression levels and perform restoration on both color and grayscale images. Second, learning compression artifact reduction models in multiple domains allows us to explore the spatial redundancies, frequency redundancies, and spatial-frequency redundancies at the same time to gain better restoration performance. As claimed in [58], the non-linear representation of the DCT-domain patches afforded by using a CNN can help discover and utilize redundancies in the DCT

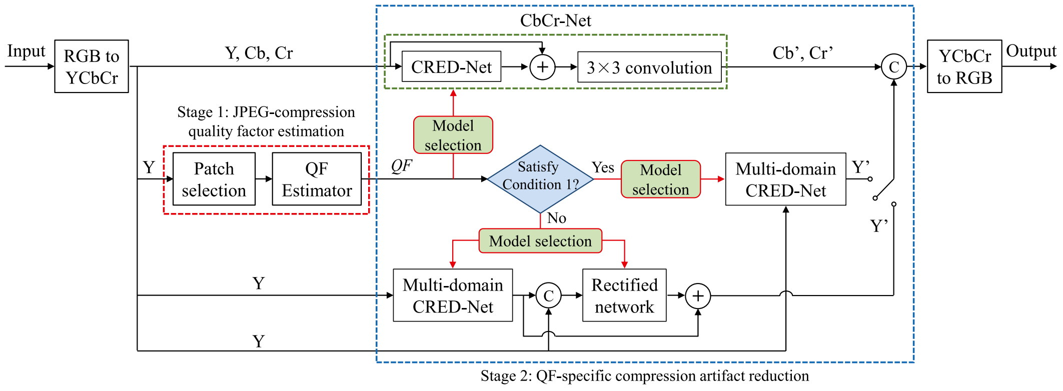

Fig. 1. A block diagram of the proposed G-CAR framework. Note that “Condition 1” denotes that the estimated QF equals one of the predefined values or falls into one of the predefined ranges. See Section 2.2.3 for more details.

domain (e.g., the inter-DCT-block correlations), but DCT coefficients mainly contain global information and do not respect the spatial continuity property of normal images. Thus, we also employ pixel and wavelet domain analyses to recover the high-frequency details, especially considering that the DWT can capture both the frequency and the location information of an image. Finally, compared with U-Net [66] and RED-Net [45] that have been widely used in image restoration and segmentation tasks, the proposed CRED-Net architecture has the advantage of learning residual feature maps at different image scales. 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 G-CAR framework. In Section 3, we analyze and discuss the performance of the proposed model on various JPEG-compressed images. General conclusions are presented in Section 4.

2. Algorithm

The goal of compression artifact reduction is to recover from a compressed image I_C an artifact free image I_R which is as similar as possible to the original uncompressed image I. Let $\mathrm{I}_C = A(\mathrm{I})$, where $A(\cdot)$ denotes a compression-decompression algorithm. Then, the problem of compression artifact reduction can been treated as to seek an inverse function $G = A^{-1}$ to satisfy $G(\mathrm{I}_C) \approx \mathrm{I}$. Different compression algorithms have different A, resulting in different compression artifacts. For JPEG, the original uncompressed image is first divided into 8×8 coding blocks, and each block undergoes the DCT transform. Then, each of the 64 DCT coefficients is quantized by referring to a 64-element quantization table controlled by a quality factor (QF). Different quantization tables can give rise to different compression levels, and it is this quantization step that produces a combination of various compression artifacts: blockiness, ringing, and blurring. As mentioned previously, the QF value of a compressed image is often unknown to a deblocking algorithm. Thus, in this paper, we propose a generalized JPEG-compression artifact reduction model, which decouples the compression artifact reduction task into two subtasks: (1) compression quality factor estimation and (2) QF-specific compression artifact reduction, as shown in Fig. 1. We provide details for each stage in the following subsections.

2.1. Compression quality factor estimation

The first stage of G-CAR is to perform JPEG-compression quality factor estimation. To this end, the Y channel of a test image is divided into non-overlapping blocks, and the QF value of each block is evaluated by a QF estimator which is a fully convolutional model. As shown in Table 1, the network consists of four types of layers: (1) convolutional layer (Conv); (2) convolutional layer followed by Rectified Linear Unit

Table 1
Network structure of the proposed QF estimator.

Layer	Kernel size	Stride	Padding	Output size
Conv	11 × 11	1	0	134 × 134 × 64
Conv	7 × 7	1	0	128 × 128 × 64
Conv+ReLU	3 × 3	1	1	128 × 128 × 128
Conv+ReLU	3 × 3	2	1	64 × 64 × 128
Conv+ReLU	3 × 3	1	1	64 × 64 × 256
Conv+ReLU	3 × 3	2	1	32 × 32 × 256
Conv+ReLU	3 × 3	1	1	32 × 32 × 512
Conv+ReLU	3 × 3	2	1	16 × 16 × 512
AvgPool	–	–	–	1 × 512
FC	–	–	–	1 × 1024
FC	–	–	–	1 × 1024
FC	–	–	–	1 × 1

(ReLU) for nonlinearity (Conv + ReLU); (3) average pooling layer (AvgPool); and (4) fully connected (FC) layer. For the first and second convolutional layers, 64 convolution filters of size 11 × 11 × 1 and 7 × 7 × 64 are respectively used to generate 64 feature maps. Then we apply six Conv+ReLU layers using a kernel size of 3 × 3 pixels and one pixel padding for high-level feature extraction. Next, average pooling is applied to collapse each feature map into a scalar, and two fully connected layers which respectively have 512 and 1024 nodes are subsequently used. The last layer is a simple linear regression with a one dimensional output that gives the QF value. Given an image patch of size 144 × 144 pixels, the dimensions of the output of each layer are shown in Table 1. The network is trained on a large number of image patches with random compression levels using L1 loss, which is computed over the predicted and ground-truth QF values.

After obtaining the QF value of each patch, a pooling operation has to be applied to collapse all values into a scalar. As in most cases the flat region of an image cannot represent JPEG compression quality properly (because JPEG compression often eliminates high-frequency details of an image while flat regions mainly consist of low-frequency components), patches with sharper edges/textures are selected. To this end, we compute local standard deviation (LSD) of the image, and then select image patches whose average LSD values are between the top 20% and top 50% of all the patches considered for QF estimation.

Specifically, the LSD of a grayscale image $I(i,j)$ is computed by

$$\sigma(i,j) = \sqrt{\sum_{k=-K}^{K} \sum_{l=-L}^{L} \omega_{k,l}(I_{k,l}(i,j) - \mu(i,j))^2}, \tag{1}$$

where

$$\mu(i,j) = \sum_{k=-K}^{K} \sum_{l=-L}^{L} \omega_{k,l} I_{k,l}(i,j) \tag{2}$$

and $\omega_{k,l}(k = -K,\dots,K; l = -L,\dots,L)$ is a 2D circularly-symmetric Gaussian weighting function sampled out to 1.5 standard deviations

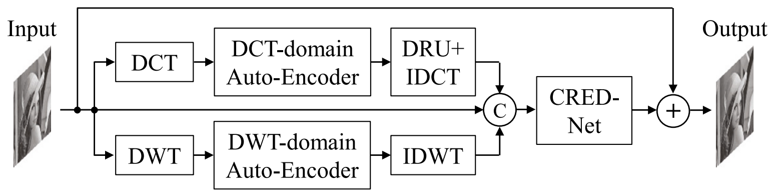

Fig. 2. Architecture of the multi-domain CRED-Net.

and rescaled to unit volume; $I_{k,l}(i,j) = I(i+k, j+l)$ denotes the local image pixel value; K and L represent the normalization window sizes. As in [62], we also define $K = L = 5$. Let QF_i $(i = 1, 2, \dots, N)$ denote the estimated QF value of each patch, and N denote the total number of all the selected patches. Then, the QF value of the overall image is computed by

$$QF = \text{Round}\left(\frac{1}{N}\sum_{i=1}^{N} QF_i\right), \tag{3}$$

where Round(·) denotes a rounding function.

2.2. QF-specific compression artifact reduction

The second stage of G-CAR is to perform the QF-specific compression artifact reduction. To this end, we propose an end-to-end multi-domain CRED-Net whose overall architecture is shown in Fig. 2. As shown in Fig. 2, a compressed image is mapped to an artifact free image via a DCT-domain auto-encoder, a DWT-domain auto-encoder, a CRED-Net, and a skipped summation connection which transforms the connectivity of the input and output to allow the residual learning. The DCT-domain auto-encoder attempts to recover the DCT coefficients of the ground truth, while the DWT-domain auto-encoder aims to restore the high-frequency details. The CRED-Net combines the two branches as well as the input compressed image to generate the residuals. Note that both the DCT and DWT domain auto-encoders share the same network structure; the only difference is the dimensions of the input and output. Also note that an additional CRED-Net is employed for further rectification when the estimated QF neither equals any predefined values nor falls into any predefined ranges (as shown in Fig. 1). We provide details of each component as follows.

2.2.1. DCT/DWT-domain auto-encoder

The architecture of the auto-encoder network is shown in Fig. 3, and the same structure is applied for both the DCT and DWT coefficients. As shown, the proposed auto-encoder contains one convolutional layer as the encoder, one convolutional layer as the decoder, and three dilated convolution layers in the middle for feature extraction in a larger receptive field. Both the encoder and decoder layers contain convolutions with 3 × 3 filters using one-pixel stride and one-pixel padding. The three dilated convolution layers also contain convolutions with 3 × 3 filters using one-pixel stride, but with different padding and dilation factors. Each convolutional layer is followed by a parametric rectified linear unit (PReLU) layer, and the number of convolutional filters for all layers in the auto-encoder is 64.

A dilated convolution applies the same filter at different ranges using different dilation factors, and has the advantage of supporting exponentially expanding receptive fields without losing resolution or coverage. For example, combining n discrete 3 × 3 filters can reach a $(2^{n+1}-1) \times (2^{n+1}-1)$ receptive field size if the dilation factors are set to be 1, 2, 4, $\cdots, 2^{n-1}$, respectively. After being first presented in [67], dilated convolution has been widely used in many vision tasks such as image segmentation (e.g., [68–70]), super-resolution (e.g., [71–73]), denoising (e.g., [74–76]), and object detection (e.g., [77–79]), etc. In this work, we set the dilation factors as 2, 4, and 8 for the three dilated convolution layers, respectively, to capture more compression artifact information of the compressed image.

For the DCT branch, a compressed image is first divided into a set of overlapping 8 × 8 blocks, and the DCT coefficients of each block are processed by the DCT-domain auto-encoder. As claimed in [38], extracting overlapping blocks at arbitrary positions that misalign with DCT coding block boundaries is very important in destroying the artificial block structures of JPEG compression method and thus effective in removing the notorious DCT blocking artifacts. To apply the DCT on overlapping blocks, in this work, a sliding window is moved across the image (or image patch) horizontally and vertically with one pixel stride, and image regions outside the window are cut off. As illustrated in Fig. 4, starting from the top-left pixel of the image, the window moves 64 times in an 8 × 8 squared manner, which results into 64 cropped images. Note that the height and width of the window should be integer multiples of eight to accommodate the 8 × 8 DCT, and also be large enough to cover most of the image. In this work, given an image of $W \times H$ pixel size, the window size is set to be $(\lfloor W/8 \rfloor \cdot 8 - 8) \times (\lfloor H/8 \rfloor \cdot 8 - 8)$, where $\lfloor \cdot \rfloor$ denotes the round-down operation. Then, the DCT is applied to each cropped image and the associated 64 DCT coefficient maps are concatenated and fed into the auto-encoder.

Following [57,58], a DCT Rectify Unit (DRU) is employed to constrain the DCT coefficient values by referring to the JPEG-specific priors. Specifically, let $x(u,v)$ denote the DCT coefficient of a compressed image, and $y(u,v)$ denote the corresponding DCT auto-encoder output, where u and v are indices in the DCT domain. Then, the rectified DCT coefficient $\tilde{y}(u,v)$ can be estimated by

$$\tilde{y}(u,v) = \begin{cases} y(u,v) - Q(u,v)/2, & y < x(u,v) - Q(u,v)/2 \\ y(u,v) + Q(u,v)/2, & y > x(u,v) + Q(u,v)/2 \\ y(u,v), & otherwise \end{cases} \tag{4}$$

where Q is the quantization table. Finally, the rectified output of the DCT auto-encoder is transformed back to the pixel domain by an inverse DCT (IDCT).

For the DWT branch, four filters (f_{LL}, f_{LH}, f_{HL}, f_{HH}) corresponding to the Daubechies 3 (db3) wavelet are convolved with the compressed image and the convolution results are then downsampled to obtain the four subband images. The wavelet subband coefficients are then concatenated and fed into the DWT-domain auto-encoder for high-level feature extraction. By concatenating the four wavelet subbands, the high-frequency components corresponding to the three different orientations are fused with the low-pass filtered image while still keeping the spatial consistency among them. Finally, a 2D inverse DWT (IDWT) is performed on the four feature maps output by the auto-encoder to produce the wavelet domain estimation, which is of the same dimension as the input image.

The DCT-domain and DWT-domain branches run in parallel. To better combine their capabilities as well as to exploit the pixel-domain redundancies, the output feature maps of the two branches and the original input image are concatenated. Note that given a $W \times H$ input image, the DCT branch outputs $(\lfloor W/8 \rfloor \cdot 8 - 8) \times (\lfloor H/8 \rfloor \cdot 8 - 8) \times 64$ feature maps. Thus, the IDWT output and the original input have to be cropped to the same width and height as the IDCT output before concatenation. In this paper, we simply crop the top-left $(\lfloor W/8 \rfloor \cdot 8 - 8) \times (\lfloor H/8 \rfloor \cdot 8 - 8)$ pixels of the IDWT output and the original input image, and the concatenated 66-channel tensor are then fed into the CRED-Net for further processing.

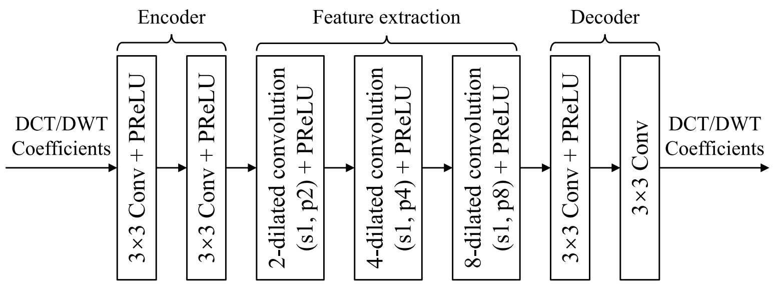

Fig. 3. Architecture of the auto-encoder network. Note that “sx” denotes *stride* = x, and “px” denotes *padding* = x.

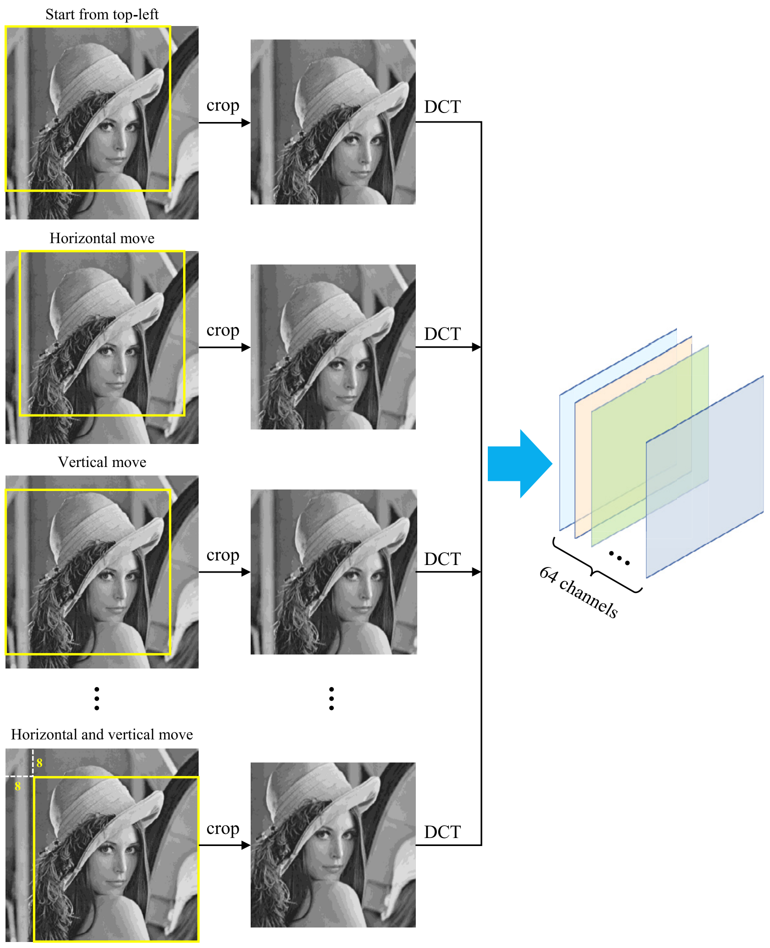

Fig. 4. Illustration of applying DCT on overlapped image blocks.

2.2.2. *CRED-Net*

The network architecture of CRED-Net is shown in Fig. 5 which consists of two subnetworks; each subnetwork contains convolutional and deconvolutional layers in the encoder and decoder with shortcut connections. The PReLU layer comes after each convolutional layer to introduce nonlinearity to the network, and a 2×2 average pooling operation with stride 2 is applied after PReLU for downsampling. Fractionally-strided convolutions (or namely “deconvolution”) are applied to the feature maps in every downsampled layer in the encoder, and the upsampled output is processed by two successive convolutional layers after being added by the high-resolution features from the scale-by-scale contracting path. For feature maps in a specific downsampled layer in the encoder, the deconvolution operation repeats the same number of times as that of downsampling. Thus, the decoder outputs feature maps containing multi-scale information in the encoder. These feature maps are concatenated and fed into a subsequent network as the input, and in this way the two subnetworks are cascaded.

Apart from the first and last layers that contain convolutions with 9×9 and 5×5 filters, respectively, all other layers contain convolutions with 3×3 filters using one-pixel stride and one-pixel padding to keep the dimension consistent with the previous feature maps. Consequently, the two subnetworks in CRED-Net can be formulated in the same way as follows. Let $x^{i,j}$ denote the output of each double-convolution operation in the first subnetwork, where i indexes the downsampling layer along the encoder and j indexes the upsampling layer along the decoder, as shown in Fig. 5. The stack of feature maps represented by $x^{i,j}$ is computed as

$$x^{i,j} = \begin{cases} F\left(x^{i-1,j}\right), & j = 0 \\ F\left(x^{i,0} + U(x^{i+1,j-1})\right), & j > 0 \end{cases} \tag{5}$$

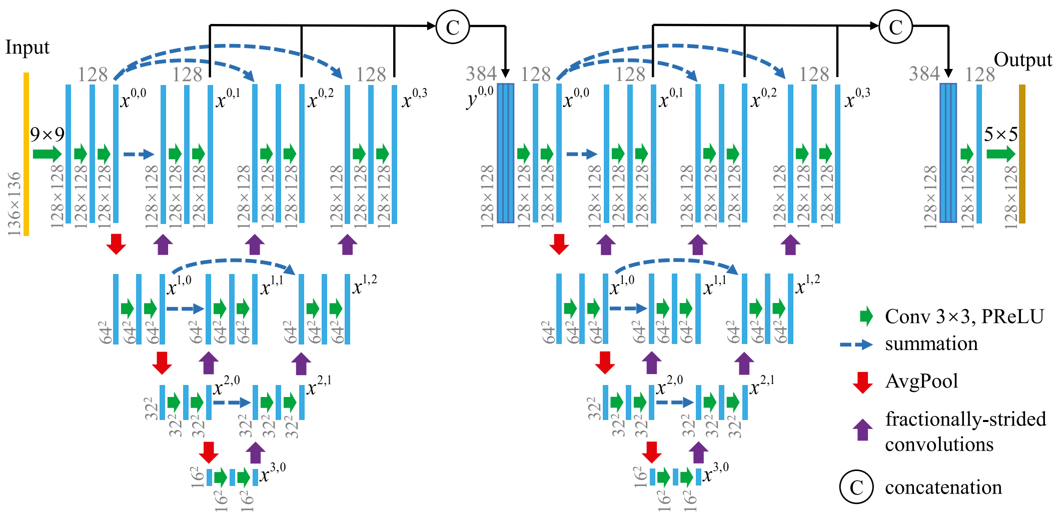

Fig. 5. An architecture of the CRED-Net.

where $F(\cdot)$ denotes the two 3×3 convolution operations each followed by PReLU; and $U(\cdot)$ denotes the 2×2 deconvolution operation using two-pixel stride and zero padding. The input feature maps of the second subnetwork (denoted by $y^{0,0}$) can be computed by

$$y^{0,0} = F\left(\left[x^{0,j}\right]_{j=1}^{3}\right) \tag{6}$$

where $[\cdot]$ denotes the concatenation operation. The output of the convolutional layers in the second subnetwork can be computed the same way as in Eq. (5). Accordingly, given a $144 \times 144 \times 1$ training image patch, after the patch is processed by the aforementioned DCT/DWT-domain auto-encoder, the input of CRED-Net is $136 \times 136 \times 66$, and the dimension of each layer output in CRED-Net is illustrated in Fig. 5. The last layer uses convolution with 5×5 filters with two-pixel refection padding, and thus the output is $128 \times 128 \times 1$. Note that although we analyze CRED-Net by assuming an input of $144 \times 144 \times 1$ pixels, images of any size can be processed by the network as long as the width and height of the input are an integral multiple of eight. Thus, in the testing stage, images are cropped with the rightmost and bottom pixels abandoned (instead of being resized) in order to meet the shape requirement as well as to preserve the original compression artifacts.

Although the shape of each subnetwork in CRED-Net is similar to U-Net [66], CRED-Net modifies U-Net in three aspects to achieve better deblocking performance: (1) Instead of fusing feature maps from the encoder and decoder via concatenation/stacking generally adopted in U-Net, we use residual connections which can help avoid the gradient vanishing problem during training when the network goes deep. (2) Unlike U-Net whose decoder has only one branch of output, each subnetwork in CRED-Net has three branches before concatenation in the decoder, where different branches represent information collected from different layers in the encoder corresponding to different image scales. As claimed in [80], image restoration is a low-level task which is supposed to require both a relatively shallow network for detail preservation and a deep network to describe the complicated relationship between the input and output. The proposed CRED-Net meets both requirements by adopting a multi-scale U-Net structure and a residual connection strategy; (3) PReLU is applied in the network instead of ReLU to avoid the dead ReLU problem in some extreme situations.

2.2.3. Compression artifact reduction

As mentioned in Section 1, most existing deblocking algorithms require the encoding parameter of a compressed image to operate effectively. To release this dependence, we employ the strategy in [55] which (1) predicts the encoding parameter first by utilizing a QF estimator, and then (2) selects the appropriate model to perform the deblocking task. To this end, we train multi-domain CRED-Nets corresponding to seven predefined QF values (5, 10, 20, 30, 40, 60, and 80). As shown in Fig. 6 in Section 3.3, the restoration performance of using any individual multi-domain CRED-Net will drop when input images are compressed with QF values different from the seven predefined values. Also, such a performance drop is significant when the QF value is small (e.g., QF = 5), but insignificant when the QF value is large (e.g., QF = 40). This is due to the fact that for smaller QF, even a minor QF variation (e.g., 5 ± 1) can cause a major change in the compression artifact intensity, while this is not the case for larger QF values.

To build a generalized compression artifact reduction framework that can effectively deal with any JPEG-compressed image, we propose to additionally train the rectified network (R-Net) whose main task is to perform fine rectification such that the performance drop observed in Fig. 6 can be compensated. Specifically, we train eight rectified networks corresponding to the eight QF ranges: $[1,4]$, $[6,7]$, $[8,9]$, $[11,14]$, $[15,19]$, $[21,24]$, $[25,29]$, and $[86,95]$. These ranges were chosen based on QF values of the cross points of the neighboring curves in Fig. 6. When the estimated QF falls into the range $[30,85]$, we simply employ a single multi-domain CRED-Net trained on the nearest predefined QF value to give the result directly, because the performance drop is acceptable as compared with the increased model complexity. Accordingly, the models selected for dealing with different Y-channel compressed images are listed in Table 2, in which QF_{est} denotes the estimated QF value of the image, and the QF value in bracket indicates the compression level of images on which the network was trained.

The rectified network shares almost the same architecture as that of CRED-Net with the following differences: (1) the dimension of the network input is $128 \times 128 \times 2$ instead of $136 \times 136 \times 66$, and accordingly there is a four-pixel padding in the first 9×9 convolution layer to keep the dimensions consistent; (2) the number of average pooling operations in the encoder has been increased to four to deal with the more difficult restoration task caused by various compression levels. The training data of the network consists of distorted image patches which are concatenated as one $128 \times 128 \times 2$ tensor and the corresponding reference patch set as the target. The two-channel tensor contains one channel from the output of the corresponding multi-domain CRED-Net and the other channel from the original input. By exploring both the original input and the intermediate deblocking result, the rectified network is able to produce decent rectifications for various levels of compression.

To enable our model to work for color images, we also train networks to perform restoration tasks on Cb and Cr channels. As shown in Fig. 1, the proposed CbCr-Net consists of a residual CRED-Net which takes all the three channels (Y, Cb, and Cr) as input, and outputs a

Table 2
Models selected for dealing with Y channel images of different compression levels.

Compressed image	Model selected
$\mathrm{QF}_{est} \in [1,4]$	Multi-domain CRED-Net (QF = 5) + R-Net (QF $\in [1,4]$)
$\mathrm{QF}_{est} = 5$	Multi-domain CRED-Net (QF = 5)
$\mathrm{QF}_{est} \in [6,7]$	Multi-domain CRED-Net (QF = 5) + R-Net (QF $\in [6,7]$)
$\mathrm{QF}_{est} \in [8,9]$	Multi-domain CRED-Net (QF = 10) + R-Net (QF $\in [8,9]$)
$\mathrm{QF}_{est} = 10$	Multi-domain CRED-Net (QF = 10)
$\mathrm{QF}_{est} \in [11,14]$	Multi-domain CRED-Net (QF = 10) + R-Net (QF $\in [11,14]$)
$\mathrm{QF}_{est} \in [15,19]$	Multi-domain CRED-Net (QF = 20) + R-Net (QF $\in [15,19]$)
$\mathrm{QF}_{est} = 20$	Multi-domain CRED-Net (QF = 20)
$\mathrm{QF}_{est} \in [21,24]$	Multi-domain CRED-Net (QF = 20) + R-Net (QF $\in [21,24]$)
$\mathrm{QF}_{est} \in [25,29]$	Multi-domain CRED-Net (QF = 30) + R-Net (QF $\in [25,29]$)
$\mathrm{QF}_{est} \in [30,34]$	Multi-domain CRED-Net (QF = 30)
$\mathrm{QF}_{est} \in [35,49]$	Multi-domain CRED-Net (QF = 40)
$\mathrm{QF}_{est} \in [50,69]$	Multi-domain CRED-Net (QF = 60)
$\mathrm{QF}_{est} \in [70,85]$	Multi-domain CRED-Net (QF = 80)
$\mathrm{QF}_{est} \in [86,95]$	Multi-domain CRED-Net (QF = 80) + R-Net (QF $\in [86,95]$)

two-channel feature map representing the restored Cb and Cr channels via a 3×3 convolution operation. The purpose of incorporating the Y channel as input is to make use of the structure/texture information in luminance to help guide the restoration of the color components. Specifically, we train eight CbCr-Nets corresponding to eight QF ranges: $[1,5]$, $[6,10]$, $[11,20]$, $[21,30]$, $[31,40]$, $[41,55]$, $[56,75]$, and $[76,95]$, and the optimal CbCr-Net is chosen based on the range into which the estimated QF value falls. Different from that being used for recovering the Y channel image, the CRED-Net used for the Cb and Cr channels output 64 (instead of 128) feature maps in each convolution layer to save model complexity, as the Cb and Cr components are less perceptible.

2.2.4. *Loss function*

As mentioned previously, the goal of compression artifact reduction is to recover from a compressed image I_C an artifact free image I_R which is as similar as possible to the original uncompressed image I. Thus, the network is learned to minimize the difference between I_R and I, which can be computed as a loss function. By referring to the previous deblocking models, we adopt both pixel-wise mean square error (MSE) loss and structural similarity (SSIM) [62] loss in our work.

The pixel-wise MSE loss is defined as

$$l_{MSE} = \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}\left[I(i,j) - I_R(i,j)\right]^2, \tag{7}$$

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 pixel-wise MSE loss is effective in recovering the lower-frequency component of an image, and thus is prone to cause blur in restoration. To combat this issue, we additionally use SSIM loss defined as

$$l_{SSIM} = 1 - \overline{\mathrm{SSIM}\left(\mathrm{I},\mathrm{I}_R\right)}, \tag{8}$$

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

$$\mathrm{SSIM}(\mathrm{I},\mathrm{I}_R) = \frac{\left(2\mu_{\mathrm{I}}\mu_{\mathrm{I}_R} + C_1\right)\left(2\sigma_{\mathrm{I}}\sigma_{\mathrm{I}_R} + C_2\right)}{\left(\mu_{\mathrm{I}}^2 + \mu_{\mathrm{I}_R}^2 + C_1\right)\left(\sigma_{\mathrm{I}}^2 + \sigma_{\mathrm{I}_R}^2 + C_2\right)}, \tag{9}$$

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 [62]. It has been demonstrated that SSIM achieves better quality assessment performance than MSE, meaning that it is more consistent with human visual perception of image similarity.

The overall loss function is a linear combination of the aforementioned two losses:

$$L = l_{MSE} + \lambda \cdot l_{SSIM}, \tag{10}$$

where $\lambda = 0.005$ is a parameter used to adjust the weights of different losses. We use the loss function defined in Eq. (10) to train the seven multi-domain CRED-Nets and the eight rectified networks for the Y channel image restoration. Also, we use pixel-wise MSE loss to train the eight CbCr-Nets for the Cb and Cr channels restoration.

3. Experiments

In this section, we conduct experiments to demonstrate the effectiveness of the proposed G-CAR framework in reducing JPEG compression artifacts. We also compare the performance of G-CAR with other state-of-the-art deblocking/soft-decoding approaches.

3.1. *Implementation details*

3.1.1. *Training data*

The training data consists of 55,000 images selected from the MS-COCO database [81], and 400 images collected from the Berkeley segmentation database (BSD) [82]. Specifically, for training the QF estimator, 45,000 pristine images were selected from the training set of MS-COCO, and the Y channel of each image was compressed using a random QF value which falls into the range $[1,95]$. Note that we do not consider images whose QF values are greater than 95 in this work, as they contain distortions that are hardly perceived by human eyes. Then, we extracted non-overlapping 144×144 patches from each of the compressed Y channel images as the training data.

For training multi-domain CRED-Net, the same 45,000 MS-COCO images, as well as 400 BSD images (200 from the training set and 200 from the validation set of BSD) were used. The same standard MATLAB JPEG encoder was applied to the Y channel of the pristine image to generate its JPEG-compressed version. As a result, for each of the seven predefined QF values, 45,400 compressed images were generated. Non-overlapping 144×144 patches were extracted from both the pristine images and their corresponding compressed versions, and were used as the training data.

For training the rectified network, 10,400 pristine images were collected (10,000 images from the testing set of MS-COCO and an extra 400 from BSD) and their JPEG-compressed Y channels with varied QF values were fed into the corresponding multi-domain CRED-Net. Then, for each of the eight QF ranges, non-overlapping 128×128 patches were extracted from the corresponding network output, which along with their original compressed inputs and ground-truth, were used as the training data.

For training CbCr-Net, the same 10,000 images were selected from the testing set of MS-COCO, and the same standard MATLAB JPEG encoder was applied to the Y, Cb, and Cr channels with varied QF values to generate their JPEG-compressed versions. Consequently, for each of the eight predefined QF ranges, 10,000 compressed images with random QF values within the specific range were generated. Then, non-overlapping $128 \times 128 \times 3$ patches and their corresponding ground-truths were extracted and used as the training data.

3.1.2. *Parameter settings and network training*

Experiments were conducted by using the PyTorch framework on a workstation with a 12-core Intel Xeon 2.67 GHz CPU and an NVIDIA GeForce GTX1080Ti GPU. The network parameters were initialized via the He initializer [83] with values sampled from the uniform distribution. The leaky slopes were initialized to 0.1 for PReLU. We used the Adam algorithm [84] with an initial learning rate of 2×10^{-4} and set the exponential decay rates for the first/second moment estimate to 0.9 and 0.999, respectively. As for the other hyper-parameters of Adam, the default settings were adopted. The learning rate was scaled down by a factor of 0.75 after every 16,000 iterations for training the multi-domain CRED-Net, rectified network, and CbCr-Net with a batch size of 16, and linearly decreased to zero in 200 epochs for training the QF estimator with a batch size of 64. As a hard-to-easy transfer, we first trained our model on image patches generated with the smallest QF value/range. Then the model corresponding to a larger QF value/range was trained based on the model corresponding to the previous smaller QF value/range. The same strategy was applied to train all networks except the QF estimator. It takes about two days to train the QF estimator, two weeks to train the seven multi-domain CRED-Nets, one week to train the eight rectified networks, and two days to train the eight CbCr-Nets. Consequently, the whole G-CAR framework takes about 25 days for training.

3.2. Algorithms and performance measures

We compared the proposed G-CAR framework with several state-of-the-art deblocking algorithms: the local-edge-regeneration-based method proposed by Golestaneh et al. [85], the dual-domain soft decoding (D2SD) algorithm [38], the trainable nonlinear reaction diffusion (TNRD) model [48], and five CNN-based methods (ARCNN [43], Fast ARCNN [86], DnCNN [44], DMCNN [57], and MWCNN [59]).

Three criteria were used to measure the performance of each deblocking algorithm: (1) peak signal-to-noise ratio (PSNR), (2) PSNR-B [87], and (3) SSIM [62]. The PSNR index estimates image quality in terms of noise, while PSNR-B modifies PSNR by additionally taking into account blocking. The SSIM index operates based on similarity measurements of three elements: luminance, contrast, and structure. It has been demonstrated in [87] that PSNR-B and SSIM correlate well with subjective quality, and perform much better than PSNR when blocking artifacts are present. Each of these measures was computed between the original (pristine, uncompressed) image and its corresponding deblocked image.

3.3. Overall quantitative results

To quantitatively evaluate the performance of G-CAR, we used as testing data JPEG-distorted grayscale images generated by JPEG-compressing-decompressing pristine images in five public benchmark datasets: LIVE1 [88], CSIQ [89], BSD100 (100 images in the validation set of BSD [82]), Classic5 (*baboon*, *barbara*, *boats*, *lena*, and *peppers*), and Urban100 [90]. Specifically, for the restoration performance test with known compression quality factors, we compressed the pristine images in all five datasets by using the standard MATLAB JPEG encoder at seven QF values: 5, 10, 20, 30, 40, 60, and 80. Note that some of these QF values (e.g., 10 and 20, etc.) are taken into account by most of the existing deblocking algorithms. To test the algorithm performance on images with unknown compression quality factors, we used the JPEG-compressed images whose QF values are less than 90 in the SD-IVL database [91]. To specifically test the performance of G-CAR on images with a wide range of compression quality factors, we used the pristine images in the LIVE1 and BSD100 datasets, and compressed each of them via the same method at QF values ranging from 1 to 95 (with a step size of 1).

3.3.1. Restoration performance

Tables 3 and 4 show the three performance measures of G-CAR and other deblocking algorithms tested on the aforementioned dataset images compressed with known and unknown QF values, respectively. Also included in the two tables are the PSNR, PSNR-B, and SSIM values of the original JPEG-compressed images for reference. Results of the best-performing deblocking algorithms are bolded in Table 3. Note that D2SD was originally designed to work on square images, and thus its testing results on LIVE1, BSD100, Urban100, and SD-IVL are unavailable. Also note that ARCNN, FastARCNN, and MWCNN were originally trained on images with QF equals 10, 20, 30, and 40 only, and thus their results on other unknown QF values are not presented. The same principle is applied to TNRD and DMCNN, which were originally trained on images with QF $\in \{10, 20, 30\}$ and QF $\in \{10, 20\}$, respectively. Among all algorithms considered, only DnCNN and G-CAR were tested on SD-IVL, as both methods take into account all possible QF values.

As can be seen from Tables 3 and 4, the proposed G-CAR framework consistently provides either the best or second-best PSNR, PSNR-B, and SSIM as compared with other deblocking algorithms. Specifically, for the unknown QF case, G-CAR shows better results than DnCNN. For the known QF case, G-CAR achieves the best performance in terms of all three performance measures when tested on images with QF equals 5, 10, 60, and 80. On images with other QF values (20, 30, and 40), G-CAR achieves the best performance in terms of PSNR-B and SSIM, and competitive results with respect to the best method in terms of PSNR. This finding holds across all of the tested datasets.

To demonstrate the effectiveness of our model over the wide range of QF values, we show improvement curves of the averaged PSNR and SSIM tested on LIVE1 and BSD100 in Fig. 6 (PSNR-B curves display similar trends as PSNR). As can be observed, the performance gain of each individual multi-domain CRED-Net drops when images with different compression levels are presented, while the G-CAR framework can achieve almost equally high performance over a wide range of QF values.

3.3.2. QF Estimation performance

As we mentioned, the G-CAR framework was tested on JPEG-compressed images generated by compressing the pristine images in LIVE1 and BSD100 using QF values ranging from 1 to 95. Here, we investigate the performance of the QF estimator by examining how accurately these 95 QF values, each of which corresponds to 129 compressed images, can be predicted. Fig. 7 shows the mean (denoted by "$\times$") and maximum-minimum prediction error bars for each of the 95 QF values. Observe that our QF estimator can predict QF values of most images quite accurately with the maximal prediction error being ± 2. Note that the mean prediction error for QF = 2 is relatively larger than the others. This is due to fact that there is only a very minor difference between images compressed with QF = 1 and QF = 2, which confuses the estimator. Also note that the estimator does not perform quite as well when QF is larger than 90. This is due to the fact that sufficiently large QF values will give rise to very minor compression artifacts which confuses the estimator. Despite these potential errors, the overall performance of G-CAR is not apparently affected, because our framework relies on QF ranges, not exact QF values, to operate effectively, owning to the employed rectified network.

3.4. Representative qualitative results

In this section, we provide visual comparisons of different deblocking algorithms applied on both synthetic and real-world compressed images. For the synthetic compressed case, we selected two representative pristine images from the LIVE1 and Classic5 datasets, and the images were compressed with QF values of 10 and 20, respectively. Figs. 8 and 9 show the input JPEG-distorted images as well as the resulting deblocking results from G-CAR and the aforementioned algorithms. Also included in the two figures are the ground-truth grayscale images for reference. As can be observed, G-CAR tends to yield better results than ARCNN, TNRD, and DnCNN. Compared with MWCNN and DMCNN, G-CAR seems to do a better job in dealing with regions that contain striped lines, which might be attributed to the multi-domain and multi-scale analysis adopted in our method.

For the real-world compressed case, the input images were JPEG images downloaded from the Internet. Fig. 10 shows three webpage images, and the corresponding deblocking results obtained from the G-CAR framework. As shown in Fig. 10(a), (c), and (e), there are some minor yet annoying compression artifacts around the edge and texture areas, and our approach can alleviate these artifacts to make the images look more pleasant. Although we are able to show only a limited set of demonstrative images, overall, G-CAR shows either highly competitive or superior deblocking performance as compared to existing methods.

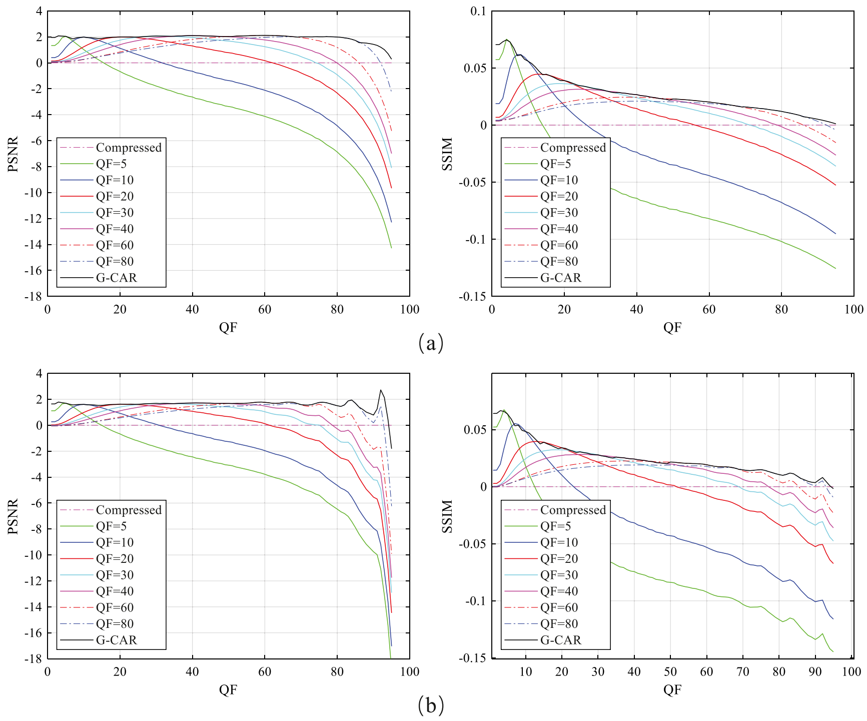

Fig. 6. Averaged PSNR and SSIM gains obtained by applying G-CAR and individual multi-domain CRED-Net on JPEG-compressed images generated from the pristine images in the LIVE1 [88] (a) and BSD100 [82] (b) databases. For each figure, the x-axis is the QF of the input and y-axis represents the corresponding performance gain where QF = N (N = 5, 10, 20, 30, 40, 60, 80) denotes the multi-domain CRED-Net specifically trained on compressed images with QF = N.

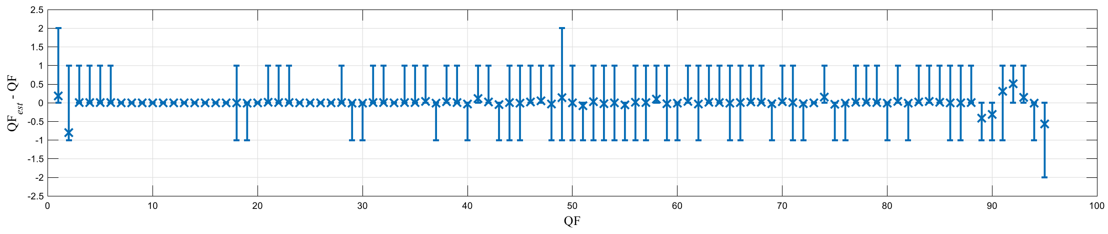

Fig. 7. Mean QF prediction errors and maximum-minimum error bars tested on LIVE1 and BSD100 dataset images compressed with a wide range of QF values. Note that the symbol "×" on each bar represents the mean QF prediction error computed over 129 images; the top and bottom sides of each bar represent the maximal and minimal QF prediction errors, respectively.

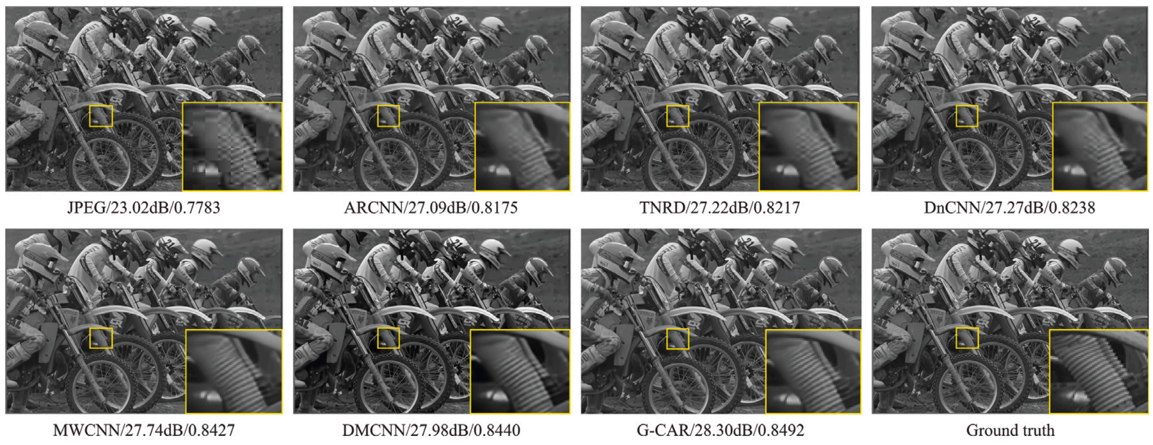

Fig. 8. Visual comparison of various deblocking methods applied on image *bikes* from the LIVE1 dataset [88] compressed with QF equals 10. The corresponding PSNR-B and SSIM values are presented at the bottom of each restored image.

Table 3
Quantitative results of the proposed G-CAR framework vs. competing methods on various images datasets measured in terms of three objective quality assessment measures averaged over the images in each dataset.

QF	Method	LIVE1 [88]			BSD100 [82]			Classic5			CSIQ [89]			Urban100 [90]		
		PSNR	PSNR-B	SSIM	PSNR	PSNR-B	SSIM	PSNR	PSNR-B	SSIM	PSNR	PSNR-B	SSIM	PSNR	PSNR-B	SSIM
5	JPEG	25.291	23.079	0.671	25.254	22.893	0.643	25.365	23.002	0.658	25.186	22.754	0.681	23.984	21.737	0.696
	Golestaneh [85]	25.402	25.201	0.702	23.399	23.129	0.664	25.755	25.407	0.699	25.443	25.077	0.708	23.761	23.566	0.718
	D2SD [38]	–	–	–	–	–	–	26.801	26.466	0.709	26.284	25.996	0.725	–	–	–
	ARCNN [43]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	FastARCNN [86]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	TNRD [48]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	DnCNN [44]	26.715	26.590	0.722	26.555	26.389	0.688	27.079	26.978	0.721	26.676	26.520	0.739	25.904	25.791	0.767
	MWCNN [59]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	DMCNN [57]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	G-CAR	**27.335**	**27.311**	**0.744**	**26.956**	**26.933**	**0.706**	**27.702**	**27.695**	**0.744**	**27.351**	**27.347**	**0.762**	**27.089**	**27.046**	**0.805**
10	JPEG	27.770	25.333	0.773	27.576	24.968	0.748	27.821	25.209	0.760	28.012	25.276	0.796	26.331	23.940	0.789
	Golestaneh [85]	27.240	27.157	0.784	24.296	24.137	0.749	26.169	26.023	0.777	27.725	27.487	0.808	25.223	25.157	0.794
	D2SD [38]	–	–	–	–	–	–	29.203	28.871	0.796	29.058	28.786	0.823	–	–	–
	ARCNN [43]	28.958	28.684	0.808	28.739	28.382	0.778	29.034	28.757	0.793	29.228	28.911	0.829	28.061	27.806	0.837
	FastARCNN [86]	28.927	28.481	0.809	28.745	28.234	0.780	29.034	28.509	0.795	29.215	28.655	0.831	28.040	27.511	0.839
	TNRD [48]	29.145	28.876	0.811	28.790	28.462	0.780	29.282	29.038	0.799	29.456	29.144	0.834	28.405	28.079	0.846
	DnCNN [44]	29.195	28.901	0.812	28.839	28.440	0.783	29.405	29.130	0.803	29.532	29.184	0.836	28.538	28.291	0.849
	MWCNN [59]	29.694	29.321	0.825	29.172	28.693	0.792	30.008	29.591	0.819	29.906	29.472	0.843	29.781	29.347	0.873
	DMCNN [57]	29.736	29.433	0.826	29.179	28.779	0.792	29.998	29.675	0.819	29.924	29.553	0.845	29.760	29.442	0.873
	G-CAR	**29.753**	**29.707**	**0.829**	**29.184**	**29.146**	**0.796**	**30.012**	**29.985**	**0.821**	**30.007**	**29.996**	**0.847**	**29.845**	**29.781**	**0.876**
20	JPEG	30.072	27.566	0.851	29.725	26.972	0.832	30.123	27.501	0.834	30.530	27.683	0.868	28.572	26.188	0.858
	Golestaneh [85]	28.510	28.507	0.852	25.460	25.433	0.822	27.117	27.090	0.837	29.576	29.537	0.871	27.068	27.063	0.856
	D2SD [38]	–	–	–	–	–	–	31.459	31.154	0.855	31.509	31.236	0.885	–	–	–
	ARCNN [43]	31.288	30.762	0.873	30.822	30.100	0.850	31.154	30.594	0.852	31.552	30.964	0.887	30.289	29.829	0.890
	FastARCNN [86]	30.967	30.235	0.873	30.793	29.896	0.851	31.145	30.228	0.853	31.511	30.548	0.890	30.300	29.427	0.891
	TNRD [48]	31.463	31.034	0.877	30.929	30.355	0.853	31.468	31.054	0.858	31.949	31.446	0.893	30.827	30.333	0.899
	DnCNN [44]	31.589	31.070	0.880	31.046	30.293	0.857	31.631	31.192	0.861	32.100	31.501	0.896	31.011	30.606	0.902
	MWCNN [59]	32.043	31.510	0.889	**31.356**	30.549	0.863	32.158	31.524	0.870	32.476	31.847	0.900	32.246	31.609	0.917
	DMCNN [57]	**32.081**	31.502	0.888	31.323	30.519	0.863	32.124	31.497	0.869	32.442	31.752	0.901	32.055	31.338	0.915
	G-CAR	32.070	**31.998**	**0.890**	31.353	**31.282**	**0.866**	**32.172**	**32.108**	**0.871**	**32.503**	**32.487**	**0.902**	**32.265**	**32.151**	**0.919**
30	JPEG	31.407	28.923	0.885	30.982	28.224	0.869	31.484	28.939	0.867	31.967	29.138	0.898	30.001	27.691	0.890
	Golestaneh [85]	29.900	29.900	0.885	26.245	26.243	0.858	28.849	28.849	0.867	31.286	31.285	0.900	29.058	29.058	0.890
	D2SD [38]	–	–	–	–	–	–	32.786	32.409	0.881	32.956	32.669	0.911	–	–	–
	ARCNN [43]	32.674	32.140	0.904	32.142	31.424	0.886	32.506	31.976	0.881	33.149	32.601	0.915	31.936	31.494	0.917
	FastARCNN [86]	31.896	30.987	0.900	31.895	30.728	0.882	32.246	31.122	0.878	32.746	31.466	0.914	31.476	30.302	0.913
	TNRD [48]	32.836	32.280	0.906	32.223	31.485	0.887	32.777	32.236	0.884	33.409	32.761	0.918	32.350	31.737	0.922
	DnCNN [44]	32.980	32.337	0.909	32.359	31.433	0.891	32.908	32.380	0.886	33.577	32.878	0.920	32.474	31.974	0.925
	MWCNN [59]	33.455	32.808	0.915	**32.674**	31.678	0.895	**33.434**	32.620	**0.893**	**33.986**	33.263	**0.925**	33.728	32.921	0.936
	DMCNN [57]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	G-CAR	**33.475**	**33.346**	**0.916**	32.669	**32.565**	**0.897**	33.391	**33.215**	**0.893**	**33.986**	**33.951**	**0.925**	**33.733**	**33.558**	**0.937**
40	JPEG	32.355	29.957	0.904	31.878	29.131	0.890	32.428	29.921	0.885	32.958	30.183	0.915	31.065	28.891	0.908
	Golestaneh [85]	30.522	30.522	0.902	26.292	26.292	0.875	29.051	29.051	0.882	31.796	31.796	0.915	29.722	29.722	0.906
	D2SD [38]	–	–	–	–	–	–	33.653	33.208	0.896	33.948	33.648	0.926	–	–	–
	ARCNN [43]	33.613	33.112	0.920	32.996	32.241	0.904	33.324	32.793	0.895	33.958	33.405	0.928	32.799	32.422	0.930
	FastARCNN [86]	32.944	32.080	0.920	32.835	31.648	0.906	33.150	32.075	0.897	33.664	32.500	0.930	32.586	31.670	0.930
	TNRD [48]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	DnCNN [44]	33.957	33.284	0.925	33.272	32.238	0.909	33.770	33.231	0.900	34.566	33.839	0.934	33.495	32.975	0.938
	MWCNN [59]	34.450	33.782	0.930	**33.602**	32.464	0.913	**34.266**	33.354	0.906	**35.001**	34.202	**0.938**	**34.766**	33.889	**0.947**
	DMCNN [57]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	G-CAR	**34.457**	**34.299**	**0.931**	33.594	**33.454**	**0.914**	34.223	**33.966**	**0.907**	34.989	**34.951**	**0.938**	34.746	**34.511**	**0.947**
60	JPEG	33.984	31.793	0.929	33.428	30.833	0.919	33.962	31.566	0.910	34.610	32.022	0.937	32.913	30.967	0.933
	Golestaneh [85]	30.825	30.825	0.923	26.831	26.831	0.898	28.585	28.585	0.901	32.567	32.567	0.933	30.580	30.580	0.926
	D2SD [38]	–	–	–	–	–	–	35.010	34.424	0.917	35.598	35.257	0.945	–	–	–
	ARCNN [43]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	FastARCNN [86]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	TNRD [48]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	DnCNN [44]	35.569	34.851	0.945	34.834	33.709	0.934	35.109	34.533	0.920	36.180	35.433	0.951	35.052	34.488	0.953
	MWCNN [59]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	DMCNN [57]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	G-CAR	**36.102**	**35.867**	**0.950**	**35.193**	**34.977**	**0.938**	**35.545**	**35.205**	**0.924**	**36.640**	**36.553**	**0.954**	**36.383**	**36.047**	**0.960**
80	JPEG	36.873	35.258	0.958	36.287	33.625	0.953	36.443	34.432	0.938	37.459	35.373	0.961	36.687	35.409	0.964
	Golestaneh [85]	31.214	31.214	0.946	26.394	26.394	0.918	27.722	27.722	0.920	33.389	33.389	0.952	31.525	31.525	0.951
	D2SD [38]	–	–	–	–	–	–	37.191	36.364	0.942	38.329	37.775	0.966	–	–	–
	ARCNN [43]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	FastARCNN [86]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	TNRD [48]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	DnCNN [44]	38.293	37.623	0.967	37.350	35.811	0.960	37.091	36.408	0.942	38.839	38.184	0.969	37.767	37.245	0.970
	MWCNN [59]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	DMCNN [57]	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	G-CAR	**38.861**	**38.384**	**0.974**	**37.793**	**37.372**	**0.969**	**37.681**	**36.998**	**0.955**	**39.325**	**39.039**	**0.976**	**39.178**	**38.521**	**0.980**

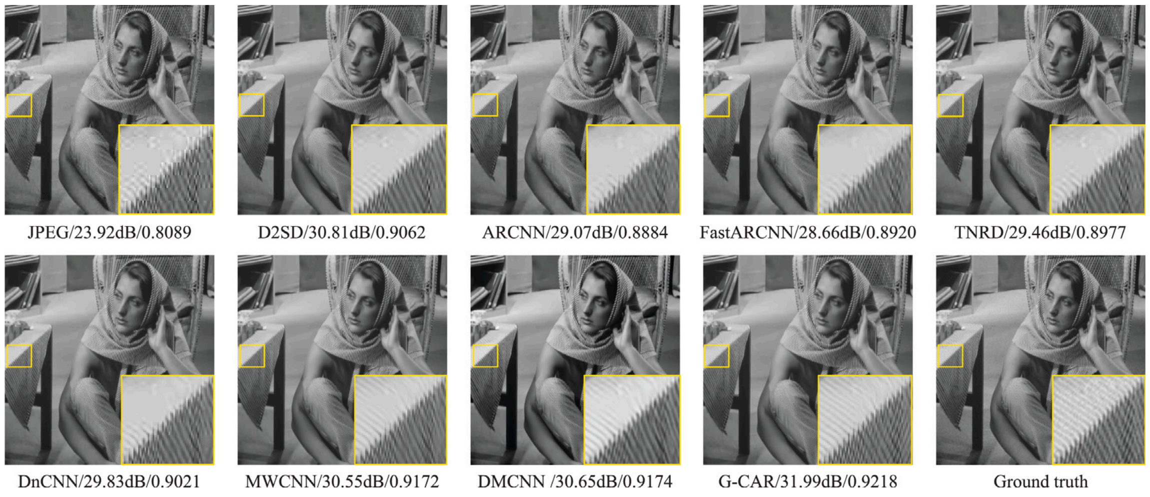

Fig. 9. Visual comparison of various deblocking methods applied on image *barbara* from the Classic5 dataset compressed with QF equals 20. The corresponding PSNR-B and SSIM values are presented at the bottom of each restored image.

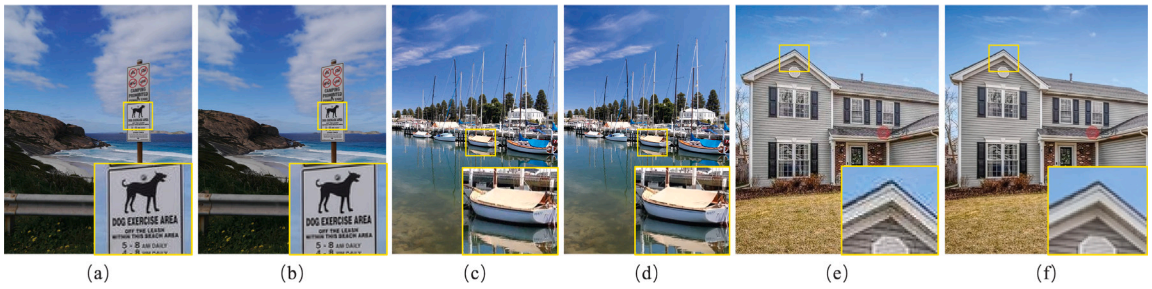

Fig. 10. Visualization of three webpage images [(a),(c),(e)] and their corresponding deblocking results [(b),(d),(f)] obtained from the G-CAR framework.

Table 4
Quantitative results of the proposed G-CAR framework vs. competing method tested on the SD-IVL database [91] measured in terms of three objective quality assessment measures averaged over images.

	PSNR	PSNR-B	SSIM
JPEG	38.129	35.220	0.949
DnCNN	39.933	38.899	0.964
G-CAR	40.045	39.960	0.967

4. Conclusion

In this paper, we presented a framework (G-CAR) for reducing artifacts in images that have undergone JPEG compression. The proposed framework consists of two stages: (1) estimation of the JPEG quality factor (QF); and (2) QF-specific artifact reduction performed via a cascaded residual encoder–decoder network in the pixel, DCT, and DWT domains. By focusing on quality-relevant regions in the QF estimation stage, G-CAR is able to quite accurately estimate the JPEG quality factor to within approximately ± 2 QF values. This estimated QF value then allows the use of QF-specific artifact reduction, assisted by trained rectified networks, when necessary. As we have demonstrated, by using multiple domains, multiple scales, and a cascaded residual encoder–decoder network architecture, the proposed framework is able to reduce JPEG artifacts at a level that rivals/surpasses the current state-of-the-art, while requiring no prior information about the encoding parameters.

Despite the effectiveness of the proposed G-CAR framework, there are a number of aspects that could benefit from future research. One line of future work would involve exploiting new network architectures that could more accurately perform the QF estimation and compression artifact reduction, or better still, to deal with different compression-level images without the assistance of the QF estimator. Other future work would involving extending G-CAR to operate for videos. Although G-CAR is effective for images, compressed videos are more widely used. Finally, more advanced frameworks that can restore images containing other distortion types (e.g., JPEG2000 compression, noise, and blur, etc.) could be developed. For example, the compression ratio (CR) of a JPEG2000 image can be first predicted by a CR estimator and then corresponding CR-specific network models could be employed to perform the deringing/deblurring task. Thus, our current work is limited to JPEG image restoration, and we envision improved artifact reduction systems that could handle multiple distortion types in future.

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.

Acknowledgments

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

References

[1] G.K. Wallace, The JPEG still picture compression standard, IEEE Trans. Consumer Electron. 38 (1) (1992) xviii–xxxiv.
[2] A. Skodras, C. Christopoulos, T. Ebrahimi, The jpeg 2000 still image compression standard, IEEE Signal Process. Mag. 18 (5) (2001) 36–58.
[3] P. Jain, V. Tyagi, A survey of edge-preserving image denoising methods, Inform. Syst. Front. 18 (1) (2016) 159–170.

[4] W.-S. Lai, J.-B. Huang, Z. Hu, N. Ahuja, M.-H. Yang, A comparative study for single image blind deblurring, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1701–1709.

[5] 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.

[6] 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.

[7] T. Meier, K.N. Ngan, G. Crebbin, Reduction of blocking artifacts in image and video coding, IEEE Trans. Circuits Syst. Video Technol. 9 (3) (1999) 490–500.

[8] L. Ma, D. Zhao, W. Gao, Learning-based image restoration for compressed images, Signal Process., Image Commun. 27 (1) (2012) 54–65.

[9] 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 Proces. 22 (12) (2013) 4613–4626.

[10] H.C. Reeve, J.S. Lim, Reduction of blocking effects in image coding, Opt. Eng. 23 (1) (1984) 230134.

[11] 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.

[12] 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.

[13] 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.

[14] B. Ramamurthi, A. Gersho, Nonlinear space-variant postprocessing of block coded images, IEEE Trans. Acoust. Speech Signal Proces. 34 (5) (1986) 1258–1268.

[15] C. Wang, J. Zhou, S. Liu, Adaptive non-local means filter for image deblocking, Signal Process., Image Commun. 28 (5) (2013) 522–530.

[16] 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.

[17] 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.

[18] Y. Luo, R.K. Ward, Removing the blocking artifacts of block-based DCT compressed images, IEEE Trans. Image Proces. 12 (7) (2003) 838–842.

[19] T. Chen, H.R. Wu, B. Qiu, Adaptive postfiltering of transform coefficients for the reduction of blocking artifacts, IEEE Trans. Circuits Syst. Video Technol. 11 (5) (2001) 594–602.

[20] S. Liu, A.C. Bovik, Efficient DCT-domain blind measurement and reduction of blocking artifacts, IEEE Trans. Circuits Syst. Video Technol. 12 (12) (2002) 1139–1149.

[21] 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.

[22] A. Nosratinia, Enhancement of JPEG-compressed images by re-application of JPEG, J. VLSI Signal Proces. Syst. Signal Image Video Technol. 27 (1–2) (2001) 69–79.

[23] R. Samadani, A. Sundararajan, A. Said, Deringing and deblocking DCT compression artifacts with efficient shifted transforms, in: 2004 International Conference on Image Processing, 2004. ICIP'04, Vol. 3, IEEE, 2004, pp. 1799–1802.

[24] 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, pp. 711–720.

[25] D. Sun, W.-K. Cham, Postprocessing of low bit-rate block DCT coded images based on a fields of experts prior, IEEE Trans. Image Proces. 16 (11) (2007) 2743–2751.

[26] T. Li, X. He, L. Qing, Q. Teng, H. Chen, An iterative framework of cascaded deblocking and superresolution for compressed images, IEEE Trans. Multimedia 20 (6) (2017) 1305–1320.

[27] 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.

[28] M. Yin, J. Gao, Y. Sun, S. Cai, Blocky artifact removal with low-rank matrix recovery, in: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2014, pp. 1996–2000.

[29] Y. Yang, N.P. Galatsanos, Removal of compression artifacts using projections onto convex sets and line process modeling, IEEE Trans. Image Proces. 6 (10) (1997) 1345–1357.

[30] 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 Proces. 25 (9) (2016) 4158–4171.

[31] 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.

[32] C. Jung, L. Jiao, H. Qi, T. Sun, Image deblocking via sparse representation, Signal Process., Image Commun. 27 (6) (2012) 663–677.

[33] H. Chang, M.K. Ng, T. Zeng, Reducing artifacts in JPEG decompression via a learned dictionary, IEEE Trans. Signal Proces. 62 (3) (2013) 718–728.

[34] 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.

[35] 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 (ICME), IEEE, 2016, pp. 1–6.

[36] 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).

[37] 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.

[38] X. Liu, X. Wu, J. Zhou, D. Zhao, Data-driven soft decoding of compressed images in dual transform-pixel domain, IEEE Trans. Image Proces. 25 (4) (2016) 1649–1659.

[39] 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 (ICIP), IEEE, 2015, pp. 306–310.

[40] 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.

[41] 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 Proces. 25 (3) (2016) 1246–1259.

[42] X. Liu, G. Cheung, X. Wu, D. Zhao, Random walk graph Laplacian-based smoothness prior for soft decoding of JPEG images, IEEE Trans. Image Proces. 26 (2) (2016) 509–524.

[43] 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.

[44] 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 Proces. 26 (7) (2017) 3142–3155.

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

[46] 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 (IJCNN), IEEE, 2017, pp. 752–759.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] X. Fu, Z.-J. 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.

[52] S. Zini, S. Bianco, R. Schettini, Deep residual autoencoder for blind universal JPEG restoration, IEEE Access 8 (2020) 63283–63294.

[53] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances In Neural Information Processing Systems, 2014, pp. 2672–2680.

[54] L. Galteri, L. Seidenari, M. Bertini, A. Del Bimbo, Deep generative adversarial compression artifact removal, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4826–4835.

[55] L. Galteri, L. Seidenari, M. Bertini, A. Del Bimbo, Deep universal generative adversarial compression artifact removal, IEEE Trans. Multimedia 21 (8) (2019) 2131–2145.

[56] 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 Proces. 2019 (1) (2019) 1–7.

[57] 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 (ICIP), IEEE, 2018, pp. 390–394.

[58] J. Guo, H. Chao, Building dual-domain representations for compression artifacts reduction, in: European Conference on Computer Vision, Springer, 2016, pp. 628–644.

[59] 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.

[60] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, 2015, arXiv preprint arXiv:1511.06434.

[61] P. Isola, J.-Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1125–1134.

[62] Z. Wang, A.C. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment: From error visibility to structural similarity, IEEE Trans. Image Proces. 13 (4) (2004) 600–612.

[63] A. Dosovitskiy, T. Brox, Generating images with perceptual similarity metrics based on deep networks, in: Advances In Neural Information Processing Systems, 2016, pp. 658–666.

[64] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, in: European Conference on Computer Vision, Springer, 2016, pp. 694–711.

[65] P. Svoboda, M. Hradis, D. Barina, P. Zemcik, Compression artifacts removal using convolutional neural networks, 2016, arXiv preprint arXiv:1605.00366.

[66] 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.

[67] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, 2015, arXiv preprint arXiv:1511.07122.

[68] Y. Wei, H. Xiao, H. Shi, Z. Jie, J. Feng, T.S. Huang, Revisiting dilated convolution: A simple approach for weakly-and semi-supervised semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7268–7277.

[69] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, G. Cottrell, Understanding convolution for semantic segmentation, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2018, pp. 1451–1460.

[70] R. Hamaguchi, A. Fujita, K. Nemoto, T. Imaizumi, S. Hikosaka, Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2018, pp. 1442–1450.

[71] W. Shi, F. Jiang, D. Zhao, Single image super-resolution with dilated convolution based multi-scale information learning inception module, in: 2017 IEEE International Conference on Image Processing (ICIP), IEEE, 2017, pp. 977–981.

[72] G. Lin, Q. Wu, L. Qiu, X. Huang, Image super-resolution using a dilated convolutional neural network, Neurocomputing 275 (2018) 1219–1230.

[73] Z. Zhang, X. Wang, C. Jung, DCSR: Dilated convolutions for single image super-resolution, IEEE Trans. Image Proces. 28 (4) (2018) 1625–1635.

[74] T. Wang, M. Sun, K. Hu, Dilated deep residual network for image denoising, in: IEEE International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2017, pp. 1272–1279.

[75] Y. Peng, L. Zhang, S. Liu, X. Wu, Y. Zhang, X. Wang, Dilated residual networks with symmetric skip connection for image denoising, Neurocomputing 345 (2019) 67–76.

[76] C. Tian, Y. Xu, L. Fei, J. Wang, J. Wen, N. Luo, Enhanced CNN for image denoising, CAAI Trans. Intell. Technol. 4 (1) (2019) 17–23.

[77] J. Li, Y. Wu, J. Zhao, L. Guan, C. Ye, T. Yang, Pedestrian detection with dilated convolution, region proposal network and boosted decision trees, in: 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, 2017, pp. 4052–4057.

[78] H. Song, W. Wang, S. Zhao, J. Shen, K.-M. Lam, Pyramid dilated deeper convlstm for video salient object detection, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 715–731.

[79] J. Zhang, C. Lu, J. Wang, L. Wang, X.-G. Yue, Concrete cracks detection based on FCN with dilated convolution, Appl. Sci. 9 (13) (2019) 2686.

[80] Z. Jiang, Y. Chen, Y. Zhang, Y. Ge, F.-F. Yin, L. Ren, Augmentation of CBCT reconstructed from under-sampled projections using deep learning, IEEE Trans. Med. Imaging 38 (11) (2019) 2705–2715.

[81] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, C.L. Zitnick, Microsoft COCO: Common objects in context, 2014, pp. 740–755.

[82] 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 (ICCV), Vol. 2, 2001, pp. 416–423.

[83] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.

[84] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[85] S.A. Golestaneh, D.M. Chandler, Algorithm for JPEG artifact reduction via local edge regeneration, J. Electron. Imaging 23 (1) (2014) 013018.

[86] K. Yu, C. Dong, C.C. Loy, X. Tang, Deep convolution networks for compression artifacts reduction, 2016, arXiv preprint arXiv:1608.02778.

[87] C. Yim, A.C. Bovik, Quality assessment of deblocked images, IEEE Trans. Image Proces. 20 (1) (2010) 88–98.

[88] H.R. Sheikh, M.F. Sabir, A.C. Bovik, A statistical evaluation of recent full reference image quality assessment algorithms, IEEE Trans. Image Proces. 15 (11) (2006) 3440–3451.

[89] 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.

[90] J.-B. Huang, A. Singh, N. Ahuja, Single image super-resolution from transformed self-exemplars, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5197–5206.

[91] S. Corchs, F. Gasparini, R. Schettini, No reference image quality classification for JPEG-distorted images, Digit. Signal Process. 30 (2014) 86–100.