Building Dual-Domain Representations for Compression Artifacts Reduction

Jun Guo(✉) and Hongyang Chao

School of Data and Computer Science, and SYSU-CMU Shunde International Joint Research Institute, Sun Yat-sen University, Guangzhou, People's Republic of China
artanis.protoss@outlook.com

Abstract. We propose a highly accurate approach to remove artifacts of JPEG-compressed images. Our approach jointly learns a very deep convolutional network in both DCT and pixel domains. The dual-domain representation can make full use of DCT-domain prior knowledge of JPEG compression, which is usually lacking in traditional network-based approaches. At the same time, it can also benefit from the prowess and the efficiency of the deep feed-forward architecture, in comparison to capacity-limited sparse-coding-based approaches. Two simple strategies, i.e., Adam and residual learning, are adopted to train the very deep network and later proved to be a success. Extensive experiments demonstrate the large improvements of our approach over the state of the arts.

Keywords: Compression artifacts reduction · Dual-domain representation · Very deep convolutional network

1 Introduction

Image restoration is a classical problem in computer vision. Among various sources of image degradation, one of the most common causes is lossy image compression (e.g., JPEG [34], WebP [10] and HEVC-MSP [32]). To date, as the popularity of mobile photo apps like Instagram and rich media social networks such as Facebook, the number of images spreading on the Internet increases rapidly. Hence, lots of companies are forced to employ lossy compression for saving bandwidth and storage space. Unfortunately, to meet the bit-budget constraint, lossy compression will inevitably introduce irreversible information loss, resulting in unwanted image artifacts. Considering that performances of miscellaneous visual tasks (e.g., image segmentation [27], image boundary detection [15], and image super-resolution [7]) greatly depend on the quality of inputs images [9], how to reduce compression artifacts has attracted more and more attentions.

Let us take JPEG as an example, as it is mostly used, to understand the compression artifacts introduced by lossy compression. As the input to the JPEG encoder, original uncompressed images are grouped into 8×8 coding blocks, and each block is forwarded to take the discrete cosine transform (DCT) separately. After DCT, each of the 64 DCT coefficients is uniformly quantized in

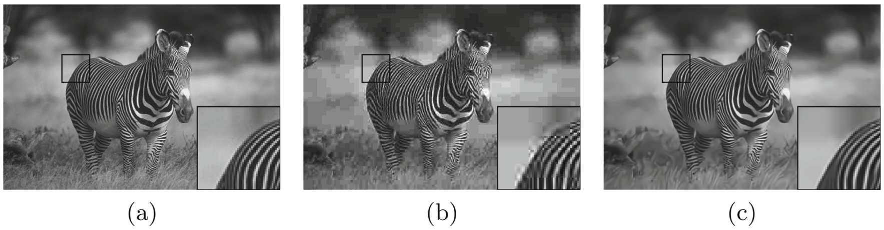


Fig. 1. (a) An uncompressed image; (b) The JPEG-compressed image of (a), where we can see multiple compression artifacts, including ringing on the boundary between the zebra and the background, blurring on the grass, and blockiness spreading on the whole image; (c) The restored image of (b) by our approach, where lots of artifacts are reduced and missing details are recovered.

conjunction with a 64-element quantization table. Note that it is this quantization step that causes a complicated combinations of numerous artifacts (see Fig. 1 for an illustration): (1) blockiness due to the individual treatment of adjacent coding blocks; (2) ringing, expressed as contours spreading along image edges, resulted from the coarse quantization of high-frequency components; and (3) blurring owing to high-frequency information loss.

Since practical lossy compression standards are not information optimal theoretically, it is possible to improve a compressed image by leveraging knowledge underused by the encoder. So far, many approaches have been proposed to deal with compression artifacts. Early works [30] tried to manually design a compression artifacts reduction procedure. Among them, pixel-domain-based approaches are rather popular. For example, Reeve and Lim [28] applied a Gaussian filter to the pixels around coding block boundaries to smooth out blocking artifacts, and Buades et al. [2] predicted pixel value by a weighted average of its surround pixels, where the weights are determined by the similarity of the corresponding image patches. Besides the pixel domain, for DCT-coded compressed images (e.g., JPEG- or HEVC-MSP-compressed images), some works tackled the problem in the DCT domain. One instance is Chen et al. [3] who applied a low pass filter to the DCT coefficients of adjacent coding blocks. Unfortunately, as aforementioned, most compression artifacts are introduced by the highly non-linear quantization step. Thus, such manually designed approaches are insufficient for modeling compression degradations and have limited restoration performances.

Nowadays, learning-based approaches are widely applied in computer vision and have achieved impressive results. The sparse coding (SC) and the convolutional neural network (CNN) are two types of the representative approaches for low-level vision tasks. For the former type [17,23,24,29], in general, input image patches are first represented by a compressed-image dictionary, and then the sparse representations (i.e., the coefficients) are passed into an uncompressed-image dictionary for reconstruction. Prior information can be naturally plugged in as constraints during the representation learning procedure. Recently, Liu et al. [24] learned sparse representations within the dual DCT-pixel domain, and achieved very promising results. However, the limited number and size of dictionaries have imposed restrictions on the capability of SC. Hence SC-based approaches tend to be accompanied with noisy edges or over-smooth regions. On the other hand, deep CNNs have been proved to possess great capability for visual tasks owing to its deep multi-layer architecture [19,31]. In particular, the SRCNN proposed by Dong et al. [7] demonstrated the power of CNNs in image restoration. Later this model was adapted to compression artifacts reduction (named ARCNN) and achieved the state-of-the-art results [6]. Nevertheless, currently CNN-based approaches just represent input images in pure pixel domain and incorporate few task-specific priors, thus there is a lot of room for improvement. Besides, as pointed out by Dong et al. [6,7], they met difficulties in training a deeper network. Transfer learning was adopted to train the four-layer ARCNN, but this technique complicates the training procedure and only obtained marginal performance gains. How to effectively train a very deep network remains to be an interesting problem.

Given the fact that JPEG images are extensive used across the world, in this paper, we still target at JPEG compression artifacts reduction. Our work tries to take advantages of both SC and CNN via a very deep architecture with JPEG-specific priors. More precisely, we build representations for input images by learning a very deep convolutional neural network in both the DCT domain and the pixel domain, namely Deep Dual-domain Convolutional neural Network (DDCN). The proposed DDCN has several appealing properties. First, learning within the dual DCT-pixel domain enables us to leverage DCT-domain prior information, so that more consistent results can be achieved, while at the same time we can still enjoy the power of traditional pixel-domain CNNs. Second, the DDCN is fully feed-forward and does not need to solve any optimization problem on usage, and hence can run much faster than most SC-based approaches like [24]. Third, with careful control of gradient updates and residual learning, the DDCN is successfully built on a much deeper architecture in comparison to previous works [6,7,24], and extensive experiments prove its superior accuracy.

2 Related Works

Our work is mainly motivated by the two state of the arts, i.e., the Dual-domain Sparse Coding (DSC) approach [24] and the ARCNN [6]. Therefore, we focus on introducing these two approaches in the following.

2.1 Dual-Domain Sparse Coding

Up to now, previous compression artifacts reduction approaches usually work either in the pixel domain [5,22,37,41,42] or in the DCT domain [8,21,39]. Unfortunately, when only operates in the pixel domain, the Inverse DCT (IDCT) is required for decompression. As a result, an isolated quantization error confined to a DCT coefficient will be propagated to all pixels of the corresponding

DCT block. What's worse, an aggressively quantized DCT coefficient can produce latent-signal-correlated structured errors in the pixel domain. Going the other way, it is extremely difficult to restore high-frequency details in pure DCT domain, since the quantization step would eliminate most high-frequency coefficients. Liu et al. [24] proposed to combine these two domains. They directly exploited residual redundancies in the DCT domain to prevent the spreading of quantization errors into the pixel domain, while at the same time a pixel-domain dictionary was learned on a large set of uncompressed images for high-frequency information recovery. In this way, the advantages of the pixel domain and the DCT domain can complement one the other. However, the DSC is overly simple (can be viewed as a two-layer network), and does not employ end-to-end training. Hence, its performance is not satisfactory, and runs extremely slow. Our work follows their dual-domain idea, but builds with a much more powerful and faster model.

2.2 Convolutional Neural Network

CNNs date back decades [20] and have recently shown explosive successes in both high-level [19,27,31] and low-level [6,7] vision tasks. Dong et al. [7] built image representations for the task of super resolution via a three-layer CNN (named SRCNN), and demonstrated impressive results. This model was further adjusted as a four-layer CNN (named ARCNN) [6] and proved to be promising in eliminating compression artifacts. However, both SRCNN and ARCNN were learned only in the pixel domain, resulting in propagation of quantization errors (as aforementioned). We will show that incorporating DCT-domain priors turns out to have great positive effect. In addition, though "deeper is better" is widely observed in high-level vision tasks, this phenomenon has not been seen in low-level ones. SRCNN and ARCNN are networks of several layers only, and they failed to obtain better performances with a deeper network. Nevertheless, we find that this problem can be somewhat mitigated by two simple tricks, i.e., gradient update restriction and residual learning.

3 Deep Dual-Domain Convolutional Network

3.1 Formulation

Consider a JPEG-compressed image $\mathbf{Y}$. Our goal is to recover from $\mathbf{Y}$ an artifact-free image $F(\mathbf{Y})$ which is as similar as possible to the original uncompressed image $\mathbf{X}$. As JPEG compression is not optimal, redundant information neglected by the JPEG encoder can still be found within a compressed image. The key issue is, how to build effective representations to automatically explore and utilize such information, so that details eliminated in the non-linear quantization step can be restored. To accomplish this task, in this paper, we develop a non-linear F as a very deep convolutional network learned in dual DCT-pixel domain, so called the Deep Dual-domain Convolutional neural Network (DDCN). Our DDCN conceptually has the following three components:

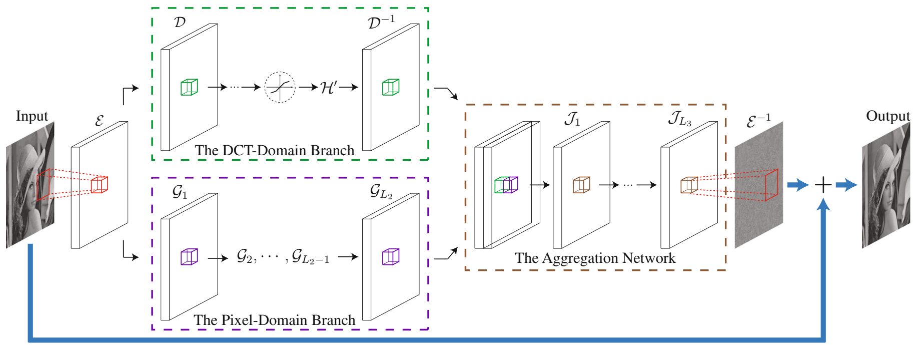


Fig. 2. An overview of the DDCN. A compressed input image is mapped to an artifact-free image via multiple non-linear layers. The DCT-domain branch tries to recover the DCT coefficients of the ground truth, while the pixel-domain branch aims to restore the pixel values directly. The aggregation network combines these two branches to produce the final output. To successfully learn a very deep model, the aggregation network predicts a residual image indeed. It is the addition of the input and the residual forms the final artifact-free output.

1. **The DCT-domain branch**: this component addresses the issue of digging DCT-domain redundancies like inter-DCT-block correlations. At the same time, DCT-domain priors such as the range of DCT coefficients are employed to improve consistency.
2. **The pixel-domain branch**: this component targets at recovering high-frequency details by leveraging spatial redundancies, e.g., similarity between image patches.
3. **The aggregation network**: this component combines the DCT-domain branch and the pixel-domain branch to generate the final artifact-free image. The prediction is expected to be similar to the uncompressed image $\mathbf{X}$.

An overview of the DDCN is depicted in Fig. 2. Without loss of generality, in the following discussions, we assume the input image $\mathbf{Y}$ is a single-channel image, i.e., a gray image. Next, we introduce each component of the DDCN in details.

3.2 The DCT-Domain Branch

To confine the quantization errors to individual DCT coefficients instead of propagating them over a wide area of pixels, we learn a non-linear mapping in the DCT domain to recover the DCT coefficients of the ground truth, namely, the DCT-domain branch. More specifically, given a JPEG-compressed image $\mathbf{Y}$, we extract a set of overlapped patches and then perform transform on them to get the corresponding DCT coefficients. Based on the resulting DCT coefficient patches, non-linear mapping modeled as a deep CNN is built, so as to automatically discover DCT-domain redundancies. This CNN is further constrained by the range of ground-truth DCT coefficients. Finally, the output is converted back

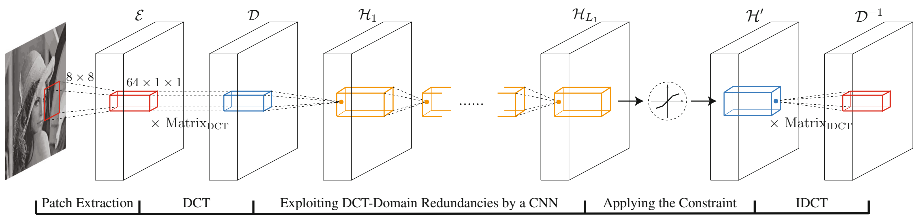


Fig. 3. A detailed illustration of the DCT-domain branch

to the pixel domain via IDCT. We detail the above operations in the following sub-sections. Figure 3 provides a summary for the DCT-domain branch.

Patch Extraction and Discrete Cosine Transform. To explore DCT-domain information of JPEG images, at first we need to extract 8 × 8 patches and then represent them by a set of pre-defined DCT bases, following the JPEG encoder. Both of these two operations can be implemented as convolution.

Formally, to extract patches of size $\sqrt{n} \times \sqrt{n}$ (for JPEG, $\sqrt{n}$ is set to 8), our first layer is expressed as an operation $\mathcal{E}$: $\mathcal{E}(\mathbf{Y}) = W_{\mathcal{E}} * \mathbf{Y}$, where $*$ denotes a convolution operator and the weights $W_{\mathcal{E}}$ contains n 2D filters of size $\sqrt{n} \times \sqrt{n}$. Precisely, $\mathcal{E}$ convolves an image with n "one-hot" filters, so that each convolution extracts one pixel of every $\sqrt{n} \times \sqrt{n}$ patch. As an example, to extract the top-left pixel of a patch, the corresponding filter can be designed as having 1 in its bottom right corner while leaving all the other elements to be 0s. Since all these filters are sparse, the operation $\mathcal{E}$ can run efficiently. Notice that although the JPEG standard works on non-overlapping blocks, we extract patches at arbitrary positions that misalign with DCT coding block boundaries. We emphasize that overlapping sampling is very important for removing artificial structures of JPEG compression, especially the notorious DCT blocking artifacts.

The output of $\mathcal{E}$ consists of n channels, where each patch is collapsed into a n-dimensional vector. Next, each patch is transformed into the corresponding DCT coefficient patch, by applying the $n \times n$ DCT matrix on it. This transform operation, say $\mathcal{D}$, can also be implemented as a convolutional layer $\mathcal{D}(\mathbf{Y}) = W_{\mathcal{D}} * \mathcal{E}(\mathbf{Y})$, where the weights $W_{\mathcal{D}}$ contains n 3D filters of size $n \times 1 \times 1$, initialized by the DCT matrix. The output of $\mathcal{D}$ is again composed of n channels.

Exploiting DCT-Domain Redundancies. So far we have converted image patches from the pixel domain to the DCT domain. To automatically discover and utilize redundancies in the DCT domain, we then extract non-linear representations from the output of $\mathcal{D}$. This non-linear operation can be formulated as a convolutional layer with a Parametric Rectified Linear Unit (PReLU) [12] applied on the filter response of convolution. To increase non-linearity, we repeat the above procedure. That is, to build layer i, we extract non-linear representations from the previous layer, and the output of layer i is further feed into the next layer to form another set of representations:

$$\mathcal{H}_i(\mathbf{Y}) = \text{PReLU}\left(W_{\mathcal{H}_i} * \mathcal{H}_{i-1}(\mathbf{Y}) + B_{\mathcal{H}_i}\right), \tag{1}$$

where $W_{\mathcal{H}_i}$ and $B_{\mathcal{H}_i}$ are the filters and biases of the i-th convolutional layer. By stacking a large number of non-linear layers, the capability of the DCT-domain branch can be significantly strengthen, so that more complicated redundancies may be exploited.

Now we have built highly non-linear representations for DCT-domain patches. To predict the ground-truth DCT coefficients, the final layer is implemented as a convolutional layer to project patch representations back to the DCT domain:

$$\mathcal{H}_{L_1}(\mathbf{Y}) = W_{\mathcal{H}_{L_1}} * \mathcal{H}_{L_1-1}(\mathbf{Y}) + B_{\mathcal{H}_{L_1}}. \tag{2}$$

Here $W_{\mathcal{H}_{L_1}}$ contains n filters, and L_1 is the number of convolutional layers in the DCT-domain branch disregarding $\mathcal{E}$, $\mathcal{D}$, and $\mathcal{D}^{-1}$ (introduced below).

Applying the Coefficient Range Constraint. The JPEG standard is composed of various pre-defined parameters. By wisely leveraging these parameters, pieces of side information can be sniffed out from the DCT coefficients of the compressed input image. As a result, JPEG-specific priors can be inferred and then applied to boost performance.

More specifically, the JPEG encoder performs quantization on a given uncompressed image as division of each DCT coefficient by its quantizer step size, followed by rounding to the nearest integer. The DCT coefficients of the corresponding compressed image are obtained by multiplying back the step size:

$$\mathsf{Y}(u,v) = \text{ROUND}(\mathsf{X}(u,v)/Q(u,v)) \cdot Q(u,v). \tag{3}$$

where X and Y are the DCT coefficients of the uncompressed image $\mathbf{X}$ and the compressed image $\mathbf{Y}$, respectively, u and v are indices in the DCT domain, and Q is the quantization table. Hence, given Y, the range of X is deterministic:

$$\mathsf{Y}(u,v) - Q(u,v)/2 \leq \mathsf{X}(u,v) \leq \mathsf{Y}(u,v) + Q(u,v)/2. \tag{4}$$

For simplicity of notation, following we denote the lower and the upper bound specified in Eq. (4) as low$(\cdot)$ and up$(\cdot)$, separately.

As discussed earlier, the DCT-domain branch tries to predict the ground-truth DCT coefficients, so the output of $\mathcal{H}_{L_1}(\mathbf{Y})$ should not exceed the corresponding DCT coefficient range. To apply this constraint, a naive thought is to add one more layer for clipping:

$$\mathcal{H}'(\mathbf{Y})_i = \begin{cases} \text{low}(\mathcal{H}_{L_1}(\mathbf{Y})_i), & \mathcal{H}_{L_1}(\mathbf{Y})_i < \text{low}(\mathcal{H}_{L_1}(\mathbf{Y})_i) \\ \mathcal{H}_{L_1}(\mathbf{Y})_i, & \text{otherwise} \\ \text{up}(\mathcal{H}_{L_1}(\mathbf{Y})_i), & \mathcal{H}_{L_1}(\mathbf{Y})_i > \text{up}(\mathcal{H}_{L_1}(\mathbf{Y})_i) \end{cases}. \tag{5}$$

Here $\mathcal{H}_{L_1}(\mathbf{Y})_i$ stands for the i-th patch in $\mathcal{H}_{L_1}(\mathbf{Y})$ (note that $\mathcal{H}_{L_1}(\mathbf{Y})$ is a set of vectors where each position is a "flattened" patch in the DCT domain), so does $\mathcal{H}'(Y)_i$. The additional layer confines the solution space and can improve the consistency and accuracy of restoration.

However, in practice the above definition of $\mathcal{H}'$ will not work. The issue, is similar to the known drawback of sigmoid in a neural network, i.e., it would make the gradients of the network become zero at almost everywhere. To overcome the vanishing gradients problem, inspired by the leaky ReLU [26], we introduce a small slope in the parts where $\mathcal{H}_{L_1}(\mathbf{Y})$ exceeds its range. As a result, Eq. (5) is rewritten as:

$$\mathcal{H}'(\mathbf{Y})_i = \begin{cases} (1-\alpha)\cdot \mathrm{low}(\mathcal{H}_{L_1}(\mathbf{Y})_i) + \alpha\cdot\mathcal{H}_{L_1}(\mathbf{Y})_i, & \mathcal{H}_{L_1}(\mathbf{Y})_i < \mathrm{low}(\mathcal{H}_{L_1}(\mathbf{Y})_i) \\ \mathcal{H}_{L_1}(\mathbf{Y})_i, & \text{otherwise} \\ (1-\alpha)\cdot \mathrm{up}(\mathcal{H}_{L_1}(\mathbf{Y})_i) + \alpha\cdot\mathcal{H}_{L_1}(\mathbf{Y})_i, & \mathcal{H}_{L_1}(\mathbf{Y})_i > \mathrm{up}(\mathcal{H}_{L_1}(\mathbf{Y})_i) \end{cases} \tag{6}$$

with $\alpha \in (0, 1]^n$ being a trainable parameter like the PReLU.

Back to the Pixel Domain. To further exploit pixel-domain redundancies, the DCT-domain reconstruction, i.e., the output of $\mathcal{H}'$, need to be transformed back to the pixel domain. Hence we define another one layer $\mathcal{D}^{-1}$ on top of $\mathcal{H}'$. Intuitively, $\mathcal{D}^{-1}$ is the inverse operation of $\mathcal{D}$, which performs IDCT on each (flatten) patch. As expected, $\mathcal{D}^{-1}$ can be also implemented as a convolutional layer, whose weights are initialized by the IDCT matrix.

3.3 The Pixel-Domain Branch

The DCT coefficients mainly contain global information of an image. That is, they do not respect the spatial continuity property of normal images. Local spatial information is tangled together in the DCT coefficients. Hence, exploiting spatial redundancies and fully recovering compressed images only in the DCT domain is not an easy task. We avoid this weakness by introducing a network directly learned in the pixel domain.

Our pixel-domain branch is straight-forward. Similar to the DCT-domain branch, at first we extract image patches using the $\mathcal{E}$ layer. After that, to automatically leverage pixel-domain redundancies, the resulting patches are directly feed into a deep CNN for non-linear representation extraction. Each layer in this CNN is of the same type: a non-linear convolutional layer decorated by the PReLU. We denote these layers as $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_{L_2}$, where L_2 is the layer number.

3.4 The Aggregation Network

The pixel-domain branch runs in parallel with the DCT-domain branch. To better combine their capability, we concatenate their outputs and then built a non-linear aggregation network to finetune the restoration results. Within the aggregation network, the predictions coming from the DCT-domain branch and the pixel-domain branch can cross validate each other, enhancing the reconstruction quality. The aggregation network is also a deep CNN, with each layer except the last containing a convolution operation followed by a PReLU nonlinearity. These layers are named as $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_{L_3}$ with L_3 being the layer number.

The last layer, denoted as $\mathcal{E}^{-1}$, operates as the inverse of the patch extraction layer $\mathcal{E}$. It puts all recovered patches back to the corresponding positions in the image. This is realized via a convolutional layer, as well.

3.5 Building Very Deep Architecture

Deep learning, as indicated by its name, uses deeper architecture to achieve higher non-linearity. The increasing non-linearity enables a network to better represent input data, resulting in higher performances. However, an opposite opinion, "deeper is not better", has been pointed out recently in certain low-level vision works [6,7].

We found that, one of the core problems is still due to exploding gradients. In high-level vision task, this problem has been largely addressed by intermediate normalization like the Batch Normalization [14]. However, such solutions are not suitable for compression artifacts reduction. As a simple example, in high-level visual task such as recognition, image contrast doesn't play a significant role, so normalization is welcomed, whereas keeping the actual contrast of an input image is very important in compression artifacts reduction and thus normalization would ruin the result. As a consequence, both [7] and [6] use an extremely small learning rate for training to avoid gradient explosion. Unluckily, in this way a deeper network is difficult to train since it will require impractically long time for convergence.

We mitigate this problem by Adam [18], a newly proposed optimization technique. This technique updates model parameters $\mathbf{\Theta}$ as $\mathbf{\Theta}_t = \mathbf{\Theta}_{t-1} - l \cdot \widehat{\mathbf{M}}_t / (\sqrt{\widehat{\mathbf{V}}}_t + \epsilon)$, where $\widehat{\mathbf{M}}$ and $\widehat{\mathbf{V}}$ are the bias-corrected first/second moment estimates of gradient, respectively, and l is the step size. As can be seen, Adam approximately restricts gradient updates by its step size hyper-parameter, so gradient explosion can be avoided in general.

Another issue is the loss function. Given a set of m uncompressed images $\{\mathbf{X}^{(i)}\}$ and their corresponding compressed images $\{\mathbf{Y}^{(i)}\}$, the Mean Squared Error (MSE) is usually adopted to learn $\{\mathbf{X}^{(i)}\}$ directly [6,7]:

$$\mathrm{Loss}(\mathbf{\Theta}) = \frac{1}{m} \sum_{i=1}^{m} \| F(\mathbf{Y}^{(i)}; \mathbf{\Theta}) - \mathbf{X}^{(i)} \|_2^2. \tag{7}$$

However, it can be seen that direct learning requires all information within $\mathbf{Y}$ being carried through the whole model F, though what we really care about is the difference between $\mathbf{X}$ and $\mathbf{Y}$. From another perspective, with the above loss function, F needs to have long-term memory for the input, which will easily lead to vanishing/exploding gradients, as pointed out by the famous LSTM [13]. Inspired by [11], we change the loss function to learn the residuals:

$$\mathrm{Loss}(\mathbf{\Theta}) = \frac{1}{m} \sum_{i=1}^{m} \| F(\mathbf{Y}^{(i)}; \mathbf{\Theta}) - (\mathbf{X}^{(i)} - \mathbf{Y}^{(i)}) \|_2^2. \tag{8}$$

It turns out that Eq. (8) results in faster convergence and higher accuracy (see Sect. 4.2). Note that, in residual learning, the DCT-domain branch is re-designed to learn the DCT coefficients of the residual image. Hence the lower bound and upper bound of $\mathcal{H}_{L_1}$ are updated to $-Q/2$ and $Q/2$, respectively.

4 Experiments

In this section, experimental results are presented to demonstrate the superior performance of the proposed DDCN for restoring JPEG-compressed images.

In all experiments, we use the training set (200 images) of the BSDS500 database [1] for training, and its validation set (100 images) for validation. Quantitative evaluations are conducted on its test set (200 images). Different from [6], all images are not down-scaled, as in this paper we don't evaluate the RTF [16] which has been beaten by the ARCNN [6]. In addition, for perceptual comparisons, we also perform qualitative evaluations on the Set14 [40] database. Following the protocol of other compression artifacts reduction approaches: (1) To generate JPEG-compressed images of different quality, the MATLAB JPEG encoder is applied with its "Quality" parameter set accordingly; (2) When reporting the restoration results, only the luminance channel (in YCbCr color space) is considered. Nevertheless, our approach is robust to JPEG encoders. A DDCN trained on MATLAB-encoded images still works well on images encoded by other JPEG encoders (e.g., the Python Image Library), with a negligible performance loss. Besides, our approach is not limited to single-channel inputs. RGB images can be easily handled by regarding each channel as a gray image.

4.1 Implementation Details

Preparing the Training Set. In the training phase, the training image pairs $\{\mathbf{Y}, \mathbf{X}\}$ are prepared as 55×55-pixel sub-images uniformly sampled from the training images with a stride of 21. We have attempted smaller strides but did not observe significant performance improvement. For a patch misaligned with DCT coding block boundaries, the DCT coefficient range of its most similar coding block is used. It should be emphasized again that overlapped sampling is important for destroying blocking artifacts.

Since the training set is small, to train the DDCN which has numerous layers, we consider augmenting the training data. More precisely, sub-images are also extracted from the randomly rotated and flipped version of the training images.

Network Settings. As specified in Sect. 3.2, n is set to 64 for JPEG compression artifacts reduction. Thus the patch extraction layer $\mathcal{E}$ contains 64 filters of size 8×8, and the transform layer $\mathcal{D}$ contains 64 filters of size $64 \times 1 \times 1$. Weights in these two layers are also initialized according to the discussions in Sect. 3.2. The settings of the inverse layers, $\mathcal{E}^{-1}$ and $\mathcal{D}^{-1}$, are similar to $\mathcal{E}$ and $\mathcal{D}$ respectively. These four layers are fixed during training.

The other layers are all composed of filters of the size $64 \times 3 \times 3$, i.e., each filter operators on 3×3 spatial regions across 64 channels. Every layer among $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_{L_1}$, $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_{L_2}$, and $\mathcal{J}_2, \mathcal{J}_3, \ldots, \mathcal{J}_{L_3}$ has 64 filters, while $\mathcal{J}_1$ contains 128 filters. If not specified, L_1, L_2, and L_3 are set to 10. Zero-padding is used before each of these convolutional layer to keep their output sizes equal. Filters are initialized using the He initializer [12] with values sampled from the Uniform distribution. The components of α in Eq. (6) are set to 0.1 initially.

To train the DDCN, we begin with a step size 0.001 and then decrease it by a factor of 10 when the validation error stops improving. The step size of Adam is reduced two times prior to termination. For the other hyper-parameters of Adam, we set the exponential decay rates for the first/second moment estimate to 0.9 and 0.999, respectively. We train a specific network with batch size 64 for each JPEG quality. During testing, the DDCN runs as a fully convolutional network [25] to generate full-image predictions.

4.2 Quantitative Evaluation on BSDS500

We examine our DDCN on the BSDS500 database, in comparison to the two aforementioned state-of-the-art approaches, i.e., the DSC [24] and the ARCNN [6]. We also include the latest generic image restoration framework, i.e., the Trainable Nonlinear Reaction Diffusion (TNRD) [4] for comparison. To have a comprehensive quantitative evaluation, experiments are conducted under three objective fidelity metrics: the PSNR (dB), the SSIM [35], and the PSNR-B (dB) [38]. Four JPEG quality settings are evaluated: 10, 20, 30, and 40.

The quantitative results are shown in Table 1. On the whole, our proposed DDCN outperforms all the state of the arts on all JPEG qualities and evaluation metrics by a large margin. Specifically, for the PSNR, we have achieved a gain of

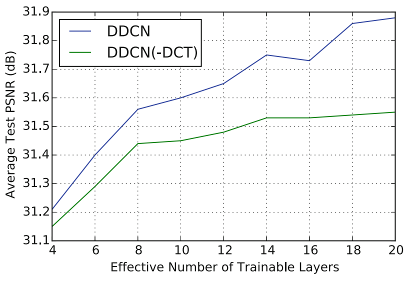
Table 1. Comparisons with the State of the Arts on the BSDS500 Database.

Quality	Evaluation	JPEG	DSC	ARCNN	TNRD	DDCN	DDCN(-DCT)
10	PSNR	27.80	28.79	29.10	29.16	**29.59**	29.26
	SSIM	0.7875	0.8124	0.8198	0.8225	**0.8381**	0.8267
	PSNR-B	25.10	28.45	28.73	28.81	**29.18**	28.89
20	PSNR	30.05	30.97	31.28	31.41	**31.88**	31.55
	SSIM	0.8671	0.8804	0.8854	0.8889	**0.8996**	0.8923
	PSNR-B	27.22	30.57	30.55	30.83	**31.10**	30.84
30	PSNR	31.37	32.29	32.67	32.77	**33.26**	32.92
	SSIM	0.8994	0.9093	0.9152	0.9166	**0.9248**	0.9193
	PSNR-B	28.53	31.84	31.94	31.99	**32.31**	32.01
40	PSNR	32.30	33.23	33.55	33.73	**34.27**	33.87
	SSIM	0.9171	0.9253	0.9296	0.9316	**0.9389**	0.9336
	PSNR-B	29.49	32.71	32.78	32.79	**33.15**	32.86

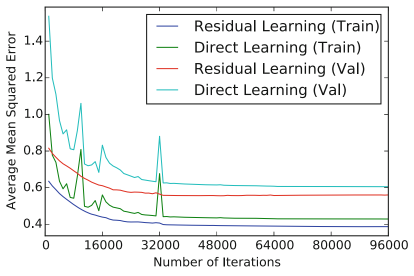
more than 0.9 dB in average compared with the DSC, and have outperformed the ARCNN by about 0.6 dB in average. The TNRD was beaten by approximately 0.5 dB in average, as well. These results demonstrate the effectiveness of our task-specific dual-domain design, and the great power of the very deep architecture. Furthermore, our DDCN also produces promising results on the PSNR-B. This metric is designed specifically to measure the degree of blockiness of a given image, and thus is more sensitive to JPEG blocking artifacts. It means our DDCN is especially suitable for improving the quality of JPEG-compressed images.

Besides, our DDCN runs much faster than the DSC, thanks to the fully feed-forward computation. For the DSC, it needs more than ten minutes to process a 512×512 image on a 6-core Xeon CPU, while the DDCN only requires several seconds. In addition, when GPUs are available, our DDCN takes less than 0.3s to generate the prediction on a GeForce GTX TITAN X, whereas the DSC employs a complicated optimization procedure and is not easy to be migrated to a GPU.

Analysis on Dual-Domain Learning. To investigate the importance of dual-domain learning, we develop a variant of the DDCN which is learned in the pixel domain only (i.e., the DCT-domain branch is dropped). Table 1 also provides experimental results for this variant (denoted as "DDCN(-DCT)"). It is clear that the PSNR values are decreased, with a loss of more than 0.3 dB in average, in comparison to the DDCN. This experiment verifies the effectiveness of leveraging DCT-domain prior information. Note that the difference in performance between the DDCN and the DDCN(-DCT) is not due to their different network sizes. We have attempted to enlarge the spatial support of all trainable filters in the DDCT(-DCT) from 3×3 to 5×5, but only observe negligible improvements (less than 0.1 dB). We have also tried to increase the depth of the DDCT(-DCT), yet turn out to be unlucky. We point out that using Eq. (5) instead of Eq. (6) made the DDCN perform almost the same as the DDCN(-DCT). Thus, it is rather significant to introduce a non-zero α for avoiding the zero gradients problem.



(a)



(b)

Fig. 4. (a) Average PSNR (dB) on the test set for various depths; (b) Average MSE on the training/validation set for residual learning and direct learning. Both experiments were conducted on BSDS500 with JPEG quality 20.

Analysis on Depth. We conduct another experiment to test the network sensitivity to different depths. For simplification, we keep layer numbers in the DCT-domain branch, the pixel-domain branch and the aggregation network to be the same value L, i.e., $L_1 = L_2 = L_3 = L$. Note that both two branches run in parallel, so the effective number of trainable layers in the DDCN is $\text{MAX}(L_1, L_2) + L_3 = 2 \cdot L$. We evaluate the DDCN under various L ranging from 2 to 10. The average PSNR values on the test set are shown in Fig. 4(a). We can see that the performance grows as the depth increases in general. Figure 4(a) also displays the performance of the DDCN(-DCT). It can be seen that, without the help of the DCT-domain branch, the DDCN(-DCT) was always inferior to the DDCN under the same layer number, and its accuracy saturated earlier. These results proved the importance of the proposed dual-domain learning again.

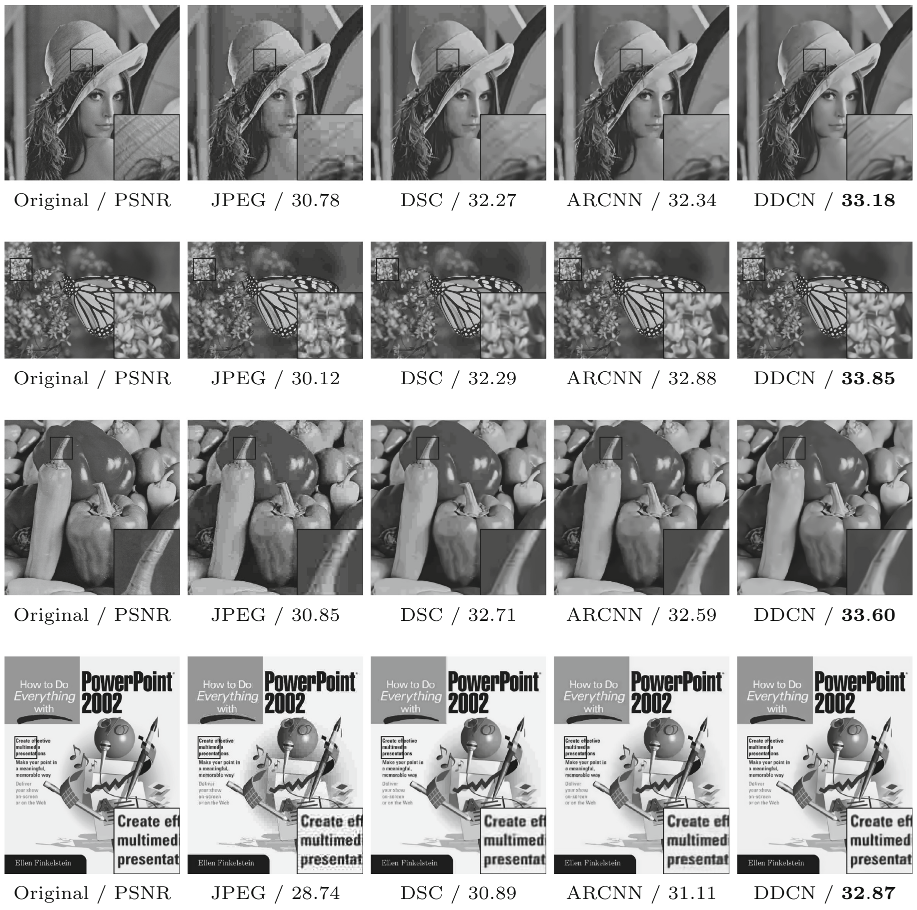


Fig. 5. Qualitative comparison of various approaches under the JPEG quality 10. The corresponding PSNR values (in dB) are also shown.

Analysis on Residual Learning. Figure 4(b) illustrates the performances of residual learning and direct learning on a 20-trainable-layer DDCN (i.e., $L = 10$). The initial step size of direct learning is selected according to the best validation loss at termination. It can be observed that, in a very deep network, learning the residuals converges much faster and obtains better performance. In addition to this, we point out that Adam is indispensable. Even given 5x training time, neither residual learning nor direct learning seemed to converge without Adam.

4.3 Qualitative Evaluation on Set14

The Set14 [40] database consist of 14 widely used test images in the literature of image processing. Here we present some restored images of different approaches on this database. As can be seen in Fig. 5, the results of the DDCN are rather visually appealing. Our approach produces much sharper edges than other approaches, without any obvious blocking or ringing artifacts across the image. This experiment demonstrates that our approach is not only superior in objective fidelity metric, but also achieves better perceptual quality.

5 Conclusions and Future Work

In this paper, we systematically studied how to build effective representations for JPEG compression artifacts reduction. Based on understanding of the JPEG compression standard, we presented a very deep convolutional network learned in both the DCT domain and the pixel domain. The dual-domain learning enabled us to incorporate DCT-domain priors naturally, while we could still benefit from the great capability of traditional pixel-domain convolutional networks. Besides, we successfully built a very deep network by two simple strategies, i.e., Adam and residual learning, and observed significant performance gain with the increasing depth. Experimental results demonstrated the promise of the proposed approach.

Our DDCN is not restricted to JPEG compression. It is worth exploring whether the DDCN is able to reduce artifacts of remaining DCT-based compression standards, such as H.264/AVC [36] and HEVC-MSP [32]. In addition, by learning in other appropriate dual-domains, e.g., wavelet-pixel domain for JPEG2000 [33], the DDCN may be even extended to remove most transform-based compression artifacts. We will leave them to future work.

One limitation of our DDCN is the scalability to various JPEG qualities. Currently we need to train a separative model for each JPEG quality, which may limit its practical usage. This restriction also exists in most state of the arts, like the ARCNN. We hope to address this issue in future.

Acknowledgment. This work is partially supported by NSF of China under Grant 61173081, 61472453, U1401256, and U1501252, and the Guangzhou Science and Technology Program, China, under Grant 201510010165.

References

1. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **33**(5), 898–916 (2011)
2. Buades, A., Coll, B., Morel, J.M.: A non-local algorithm for image denoising. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, vol. 2, pp. 60–65. IEEE (2005)
3. Chen, T., Wu, H.R., Qiu, B.: Adaptive postfiltering of transform coefficients for the reduction of blocking artifacts. IEEE Trans. Circ. Syst. Video Technol. **11**(5), 594–602 (2001)
4. Chen, Y., Yu, W., Pock, T.: On learning optimized reaction diffusion processes for effective image restoration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5261–5269 (2015)
5. Chuah, S., Dumitrescu, S., Wu, X.: l_2 optimized predictive image coding with l_∞ bound. IEEE Trans. Image Process. **22**(12), 5271–5281 (2013)
6. Dong, C., Deng, Y., Change Loy, C., Tang, X.: Compression artifacts reduction by a deep convolutional network. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 576–584 (2015)
7. Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014, Part IV. LNCS, vol. 8692, pp. 184–199. Springer, Heidelberg (2014)
8. Foi, A., Katkovnik, V., Egiazarian, K.: Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images. IEEE Trans. Image Process. **16**(5), 1395–1411 (2007)
9. Gao, W., Tian, Y., Huang, T., Ma, S., Zhang, X.: The IEEE 1857 standard: empowering smart video surveillance systems. IEEE Intell. Syst. **29**(5), 30–39 (2014)
10. Google: WebP - a new image format for the web. Google Developers Website. https://developers.google.com/speed/webp/
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015). arXiv preprint arXiv:1512.03385
12. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034 (2015)
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
14. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift (2015). arXiv preprint arXiv:1502.03167
15. Isola, P., Zoran, D., Krishnan, D., Adelson, E.H.: Crisp boundary detection using pointwise mutual information. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014, Part III. LNCS, vol. 8691, pp. 799–814. Springer, Heidelberg (2014)
16. Jancsary, J., Nowozin, S., Rother, C.: Loss-specific training of non-parametric image restoration models: a new state of the art. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part VII. LNCS, vol. 7578, pp. 112–125. Springer, Heidelberg (2012)
17. Jung, C., Jiao, L., Qi, H., Sun, T.: Image deblocking via sparse representation. Sig. Process. Image Commun. **27**(6), 663–677 (2012)
18. Kingma, D., Ba, J.: Adam: a method for stochastic optimization (2014). arXiv preprint arXiv:1412.6980

19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
20. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4), 541–551 (1989)
21. Lee, K., Kim, D.S., Kim, T.: Regression-based prediction for blocking artifact reduction in JPEG-compressed images. IEEE Trans. Image Process. **14**(1), 36–48 (2005)
22. List, P., Joch, A., Lainema, J., Bjontegaard, G., Karczewicz, M.: Adaptive deblocking filter. IEEE Trans. Circ. Syst. Video Technol. **13**(7), 614–619 (2003)
23. Liu, X., Cheung, G., Wu, X., Zhao, D.: Inter-block consistent soft decoding of JPEG images with sparsity and graph-signal smoothness priors. In: 2015 IEEE International Conference on Image Processing (ICIP), pp. 1628–1632. IEEE (2015)
24. Liu, X., Wu, X., Zhou, J., Zhao, D.: 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, pp. 5171–5178 (2015)
25. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
26. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. Proc. ICML **30**, 1 (2013)
27. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1520–1528 (2015)
28. Reeve III, H.C., Lim, J.S.: Reduction of blocking effects in image coding. Opt. Eng. **23**(1), 230134–230134 (1984)
29. Rothe, R., Timofte, R., Van, L.: Efficient regression priors for reducing image compression artifacts. In: 2015 IEEE International Conference on Image Processing (ICIP), pp. 1543–1547. IEEE (2015)
30. Shen, M.Y., Kuo, C.C.J.: Review of postprocessing techniques for compression artifact removal. J. Vis. Commun. Image Represent. **9**(1), 2–14 (1998)
31. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014). arXiv preprint arXiv:1409.1556
32. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (HEVC) standard. IEEE Trans. Circ. Syst. Video Technol. **22**(12), 1649–1668 (2012)
33. Taubman, D.S., Marcellin, M.W.: JPEG 2000: standard for interactive imaging. Proc. IEEE **90**(8), 1336–1357 (2002)
34. Wallace, G.K.: The JPEG still picture compression standard. IEEE Trans. Consum. Electron. **38**(1), 18–34 (1992)
35. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
36. Wiegand, T., Sullivan, G.J., Bjøntegaard, G., Luthra, A.: Overview of the H. 264/AVC video coding standard. IEEE Trans. Circ. Syst. Video Technol. **13**(7), 560–576 (2003)
37. Yang, Y., Galatsanos, N.P., Katsaggelos, A.K.: Projection-based spatially adaptive reconstruction of block-transform compressed images. IEEE Trans. Image Process. **4**(7), 896–908 (1995)

38. Yim, C., Bovik, A.C.: Quality assessment of deblocked images. IEEE Trans. Image Process. **20**(1), 88–98 (2011)
39. Zakhor, A.: Iterative procedures for reduction of blocking effects in transform image coding. IEEE Trans. Circ. Syst. Video Technol. **2**(1), 91–95 (1992)
40. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: Boissonnat, J.-D., Chenin, P., Cohen, A., Gout, C., Lyche, T., Mazure, M.-L., Schumaker, L. (eds.) Curves and Surfaces 2011. LNCS, vol. 6920, pp. 711–730. Springer, Heidelberg (2012)
41. Zhai, G., Zhang, W., Yang, X., Lin, W., Xu, Y.: Efficient deblocking with coefficient regularization, shape-adaptive filtering, and quantization constraint. IEEE Trans. Multimedia **10**(5), 735–745 (2008)
42. Zhou, J., Wu, X., Zhang, L.: l_2 restoration of l_∞-decoded images via soft-decision estimation. IEEE Trans. Image Process. **21**(12), 4797–4807 (2012)