JPEG Artifacts Reduction via Deep Convolutional Sparse Coding

Xueyang Fu[1], Zheng-Jun Zha[1]*, Feng Wu[1], Xinghao Ding[2], John Paisley[3]
[1]School of Information Science and Technology, University of Science and Technology of China, China
[2]School of Informatics, Xiamen University, China
[3]Department of Electrical Engineering & Data Science Institute, Columbia University, USA
{xyfu, zhazj, fengwu}@ustc.edu.cn, dxh@xmu.edu.cn, jpaisley@columbia.edu

Abstract

To effectively reduce JPEG compression artifacts, we propose a deep convolutional sparse coding (DCSC) network architecture. We design our DCSC in the framework of classic learned iterative shrinkage-threshold algorithm [16]. To focus on recognizing and separating artifacts only, we sparsely code the feature maps instead of the raw image. The final de-blocked image is directly reconstructed from the coded features. We use dilated convolution to extract multi-scale image features, which allows our single model to simultaneously handle multiple JPEG compression levels. Since our method integrates model-based convolutional sparse coding with a learning-based deep neural network, the entire network structure is compact and more explainable. The resulting lightweight model generates comparable or better de-blocking results when compared with state-of-the-art methods.

1. Introduction

Image compression methods such as JPEG, WebP and HEVC-MSP, allow for efficient image storage and transmission. However, their possibly coarse quantization can result in visible artifacts. These artifacts severely degrade both perceptual quality and subsequent computer vision systems. Reducing the artifacts of lossy compression is an important computer vision task, most notably for the most widely used JPEG format.

JPEG compression is achieved by applying a discrete cosine transformation (DCT) on 8×8 pixel blocks. The DCT coefficients are further coarsely quantized to save storage space. However, this scheme results in image discontinuities at the boundaries of blocks, which generate blocking and blurring artifacts. To improve the quality of JPEG compressed images, different methods have been explored. In general, JPEG artifacts reduction methods are either model-based [31, 46, 11, 33, 29] or learning-based [8, 41, 53, 6, 14, 55]. The former is usually designed using domain knowledge while the latter aims to learn a nonlinear mapping function directly from training data.

Though learning-based methods have achieved competitive performance compared with model-based methods for JPEG artifacts reduction, these two methods have complementary merits. Model-based methods usually have clear physical meaning, but suffer from time-consuming optimization, while learning-based methods have fast testing speed, but the learned model has insufficient explainability. Recent work has shown the advantage of combining powerful deep learning modeling approaches with model-based sparse coding (SC) for JPEG artifacts reduction via a dual-domain network [41]. In this paper, we develop this initial line of work based on shallow, fully connected layers by using ideas from convolutional sparse coding [4, 22, 36]. Since convolutional sparse coding (CSC) is spatially invariant and can directly process the whole image, it is suitable for various low-level vision tasks, such as image super-resolution [18] and layer decomposition [50, 17, 51].

Our main contribution is to propose a deep model for JPEG artifacts reduction that has increased explainability. Using the framework of learned iterative shrinkage-threshold algorithm (LISTA) for CSC, we construct a recursive deep model that isolates artifacts from the latent feature maps. We utilize dilated convolutions for multi-scale image features extraction, allowing our single model to handle multiple JPEG qualities. Since we follow the classical optimization algorithm in building our deep convolutional sparse coding, each module is specifically designed rather than simply stacked, making our model compact and easy to implement. Compared with several state-of-the-art methods, our model achieves comparable or better de-blocking performance based on both qualitative and quantitative evaluations.

*Corresponding author. This work was supported by the National Key R&D Program of China under Grant 2017YFB1300201, the National Natural Science Foundation of China (NSFC) under Grants 61622211, 61620106009 and 61701158 as well as the Fundamental Research Funds for the Central Universities under Grant WK2100100030.

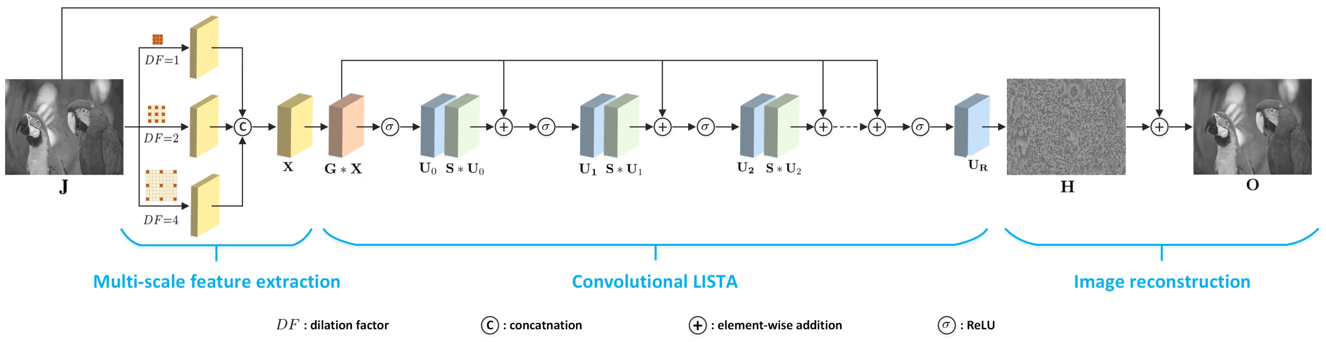


Figure 1. The framework of the proposed DCSC network for JPEG artifacts reduction. Each component of our network is designed to complete a specific task. First, dilated convolutions are performed on the JPEG compressed image $\mathbf{J}$ to obtain multi-scale features $\mathbf{X}$. Then, a convolutional and iterative LISTA is constructed to sparsely code $\mathbf{X}$ to recognize and separate artifacts. The learned sparse codes $\mathbf{U}_R$ are finally used to generate the residual $\mathbf{H}$ for predicting the de-blocked image $\mathbf{O}$. Note that $\mathbf{S}$ is shared across iterations according to the feed-forward unrolling form of ISTA (3).

1.1. Related work

Early JPEG artifacts reduction methods depended heavily on design, such as filtering in the image domain [31, 7, 46] or the transform domain [30]; for example, [11] use a shape-adaptive DCT-based image filter and [49] use a joint image/DCT domain filtering algorithm. Another direction is to formulate artifacts removal as an ill-posed inverse problem and solve via optimization, for example by projecting onto convex sets [43], using regression trees [24], image decomposition [29] or the non-local self-similarity property [52, 54, 28]. Sparsity has been fully explored as a technique for effectively regularize this ill-posed problem [5, 33, 34, 32].

In the past few years, notable progress has been made for this problem by deep learning-based methods using deep convolutional neural networks (CNNs). These methods aim at learning a nonlinear mapping from uncompressed image paired with its corresponding JPEG compressed version. The first deep CNNs-based method for JPEG artifacts reduction was introduced by [8], where the authors design a relatively shallow network structure on the basis of the popular super-resolution network [9]. In [6], a trainable nonlinear reaction diffusion model for general image restoration was proposed. Inspired by the success of residual learning [21] and dense connection [23] in high-level vision tasks, two very deep networks were introduced for general image restoration, including JPEG artifacts reduction, image denoising and super-resolution [53, 39].

Since increasing network depth can increase the receptive field, excellent performance is achieved by [53, 39]. Based on the JPEG-related priors in both image and DCT domains [34], two different dual-domain networks [19, 41] for the de-blocking are respectively proposed. In [19], the authors design a 20-layer dual-domain network to eliminate complex artifacts. In [41], to obtain speed and performance gains, the authors build a cascaded network to perform LISTA [16] in dual-domain. In [45], to effectively recover high frequency details, the authors formulate the de-blocking problem as a classification problem in the DCT domain. Inspired by the attractive generative adversarial networks [15, 27], some de-blocking methods [14, 20] are proposed to generate photo-realistic details to further improve the visual quality. More recently, a decouple learning framework [10] is proposed to incorporate different parameterized image operators for image filtering and restoration tasks. Our network architecture shares the similar spirit with method [41] by combining domain knowledge and deep learning for JPEG artifacts reduction.

2. Methodology

We show the framework of the proposed DCSC in Figure 1. As illustrated, our model contains three components: multi-scale feature extraction, convolutional LISTA on the extracted features, followed by image reconstruction. The entire network is an end-to-end system which takes a JPEG compressed image $\mathbf{J}$ as input and directly generates the output image $\mathbf{O}$. The network is fairly straightforward, with each component designed to achieve a specific task. Below we describe our network architecture and training strategy.

2.1. Network architecture

2.1.1 Multi-scale feature extraction

Because JPEG compression quality can vary as desired, artifacts due to compression can vary in their spatial extent, *e.g.*, a lower JPEG quality causes larger scale artifacts in smooth areas. To handle different JPEG qualities, existing deep learning-based methods either train individual models on each specific JPEG quality [9] or construct deep models to the increase receptive field at the cost of a larger parameter burden [53, 39]. To address this problem, we instead adopt dilated convolutions [47] to extract multi-scale features. By dilating the same filter to different scales, dilated convolutions can increase the contextual area without introducing extra parameters. Specifically, we first generate a series of features using different dilation factors, and then concatenate these features,

$$\mathbf{X}_{DF} = \mathbf{W}_{DF} * \mathbf{J} + \mathbf{b}_{DF}, \\ \mathbf{X} = concat(\mathbf{X}_{DF}), \tag{1}$$

where DF is the dilation factor, $\mathbf{X}_{DF}$ is the output feature of dilated convolution, $*$ indicates the convolution operation, $\mathbf{W}_{DF}$ and $\mathbf{b}_{DF}$ are the kernels and biases in the dilated convolution, $concat(\cdot)$ denotes the concatenation.

In this work, we use three dilation factors, *i.e.*, $DF \in \{1, 2, 4\}$, which we empirically noticed was sufficient for our problem. Figures 2(d)-(e) show visual examples of different $\mathbf{X}_{DF}$ of a JPEG compressed image (quality = 10). Clearly, as the dilation factor increases, the corresponding feature map captures larger scale structure and content. Since the concatenated $\mathbf{X}$ contains multi-scale JPEG compression features, our single network can handle different JPEG qualities.

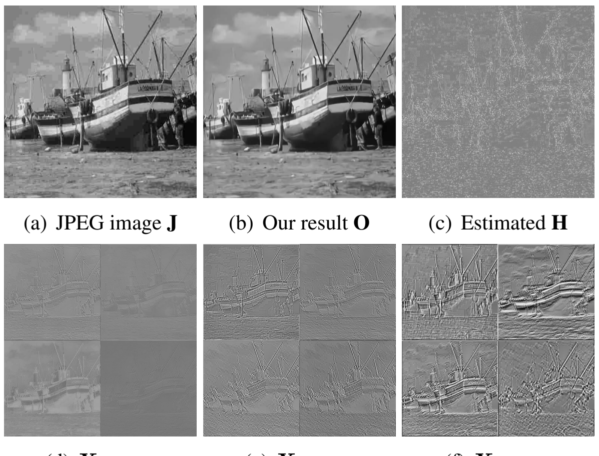

(a) JPEG image $\mathbf{J}$
(b) Our result $\mathbf{O}$
(c) Estimated $\mathbf{H}$
(d) $\mathbf{X}_{DF=1}$
(e) $\mathbf{X}_{DF=2}$
(f) $\mathbf{X}_{DF=4}$

Figure 2. Visualization of multi-scale features $\mathbf{X}_{DF}$. We only show 4 feature maps for each dilation factor for visualization.

2.1.2 Convolutional LISTA

To combine the advantages of model-based and learning-based methods for JPEG artifacts reduction, we design the second component of our network by integrating classic CSC approach [4, 51] with LISTA [16]. The problem of original sparse coding is to find the optimal sparse code that minimizes following objective function (2) with ℓ_1-regularized code $\mathbf{u}$,

$$\arg\min_{\mathbf{u}} \|\mathbf{x} - \mathbf{\Phi}\mathbf{u}\|_F + \lambda\|\mathbf{u}\|_1, \tag{2}$$

where $\mathbf{x}$ is the input signal, $\mathbf{\Phi}$ is an over-complete dictionary, $\mathbf{u}$ is the corresponding sparse code, F is the Frobenius norm, and λ is a positive parameter. To solve the SC problem (2), the original ISTA was introduced to find sparse codes with the following iterative equation,

$$\begin{aligned} \mathbf{u}_r &= \sigma_\theta(\mathbf{u}_{r-1} + \frac{1}{L}\mathbf{\Phi}^T(\mathbf{x} - \mathbf{\Phi}\mathbf{u}_{r-1})) \\ &= \sigma_\theta(\frac{1}{L}\mathbf{\Phi}^T\mathbf{x} + (\mathbf{I} - \frac{1}{L}\mathbf{\Phi}^T\mathbf{\Phi})\mathbf{u}_{r-1}) \\ &= \sigma_\theta(\mathbf{G}\mathbf{x} + \mathbf{S}\mathbf{u}_{r-1}), \\ \mathbf{u}_0 &= \sigma_\theta(\mathbf{G}\mathbf{x}), \end{aligned} \tag{3}$$

where r is the current iteration, L is a constant that must be an upper bound on the largest eigenvalue of $\mathbf{\Phi}^T\mathbf{\Phi}$, $\sigma_\theta(\cdot)$ is the shrinkage function with a threshold θ [13]. To speed up ISTA for real-time applications, LISTA [16] was proposed to approximate the sparse coding of ISTA by learning parameters from data.

Even though conventional SC has been extended to various image restoration tasks [2, 42, 5], these methods learn multiple features that are in fact shifted versions of the same feature. To address this issue, convolutional sparse coding methods have been introduced to build the objective function in a shift invariant way [51], which is achieved by

$$\arg\min_{\mathbf{w},\mathbf{U}} \left\|\mathbf{X} - \sum_{m=1}^{M} \mathbf{w}(m) * \mathbf{U}(m)\right\|_F + \lambda \sum_{m=1}^{M} \|\mathbf{U}(m)\|_1. \tag{4}$$

When M sparse coefficients $\mathbf{U}$ are convolved with M convolutional dictionaries $\mathbf{w}$, it can be used to approximate the input image $\mathbf{X}$. To solve (4), several algorithms have been proposed [17, 51] with time-consuming optimization. However, the convolution operation can be performed as a matrix multiplication by transforming the kernel into a circulant matrix [35]. Therefore, the CSC model (4) can be seen as a special case of the general SC model (2). This motivates us to modify the form of ISTA (3) as the second component of our network, where matrix multiplication was replaced by the convolutional operation.

At this time, the convolutional dictionaries in classical CSC are embedded into learnable convolutional kernels $\mathbf{G}$ and $\mathbf{S}$, the sparse feature coefficients become the feature maps $\mathbf{U}_r$. We choose the widely used Rectified Linear Unit (ReLU) [26] as the non-linear activation function $\sigma_\theta(\cdot)$ since ReLU is able to introduce sparsity into the model. Note that $\mathbf{S}$ is shared according to the iterative form. In fact, there is a coupling relationship between $\mathbf{G}$ and $\mathbf{S}$, *i.e.*, $\mathbf{S} = \mathbf{I} - \mathbf{G}\mathbf{\Phi}$ according to Equation (3). However, this relationship limits the model flexibility and capacity. To take full advantages of deep learning, we use independent kernels to individually model $\mathbf{G}$ and $\mathbf{S}$.

Figure 3 shows an example of the learned sparse features $\mathbf{U}_R$. It is clear that JPEG artifacts (top row) and object structures (bottom row) are recognized and separated. This indicates that by combining a model-based convolutional LISTA with deep learning, a simple network structure is able to learn effective and discriminative features.

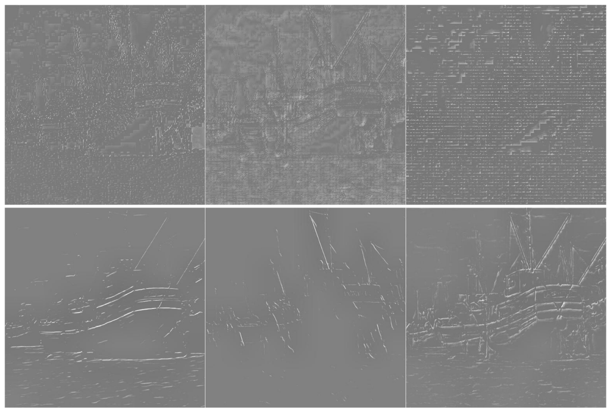


Figure 3. Visualization of learned sparse features $\mathbf{U}_R$ of Figure 2. Top row shows separated blocking artifacts and bottom row shows clear structure. Due to space limitations, we only show 6 features.

To demonstrate the explainability, below we visualize two sparse feature maps (after ReLU) during iterations. As shown in Figure 4, as the iteration progresses, the content becomes stable. This is consistent with traditional optimization methods, *i.e*., with many iterations the sparse code converges to the optimal solution. Since we can clearly observe the changing state of the feature maps, our deep CSC allows a kind of explainability to take place during inference. Meanwhile, the sparsity can also be observed in feature maps since a certain number of pixel values equals 0 after ReLU.

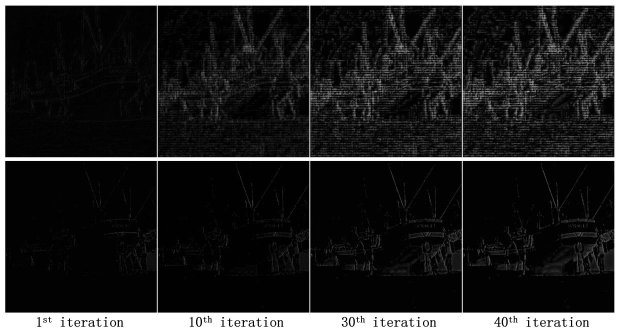


Figure 4. Examples of two sparse feature maps during iterations.

2.1.3 Image reconstruction

After R iterations, the sparse feature maps $\mathbf{U}_R$ are finally fed into a convolutional layer to generate the output image. Inspired by the methods of [53, 12], in which residual information is used to simplify the learning problem, we map $\mathbf{U}_R$ to the residual image $\mathbf{H}$,

$$\mathbf{H}=\mathbf{W}_R*\mathbf{U}_R+\mathbf{b}_R, \tag{5}$$

where $\mathbf{W}_R$ and $\mathbf{b}_R$ are the parameters in the convolution. As shown in Figure 2(c), the estimated residual $\mathbf{H}$ mainly contains blocking artifacts and high-frequency information. The final de-blocked image $\mathbf{O}$ is obtained by calculating

$$\mathbf{O} = \mathbf{J} + \mathbf{H}. \tag{6}$$

2.2. Loss function

The most widely used loss function for image restoration tasks is mean squared error (MSE) [8, 53]. However, MSE usually generates over-smoothed results due to the squared penalty that works poorly at edges within an image. Instead, we use the mean absolute error (MAE) for network training. MAE does not over-penalize larger errors and thus can preserve structures and edges, as is well known in total variation minimization applications. Given N training image pairs $\{\mathbf{O}^i, \mathbf{J}^i\}_{i=1}^N$, the goal is to minimize the following objective function,

$$\mathcal{L}(\mathbf{\Theta})=\frac{1}{N}\sum_{i=1}^{N}\left\|f(\mathbf{J}^i;\mathbf{\Theta})-\mathbf{O}^i\right\|_1, \tag{7}$$

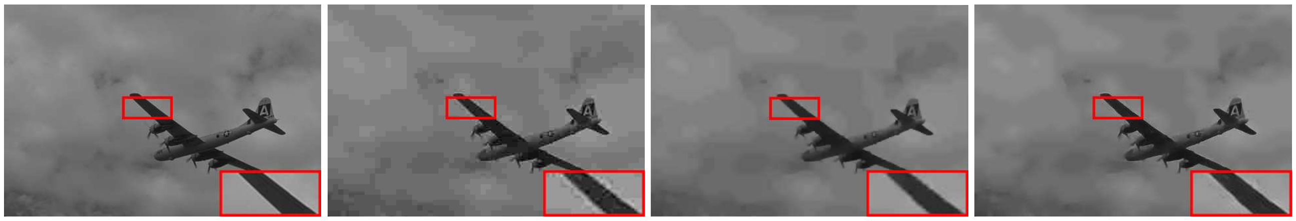
where $f(\cdot)$ denotes our DCSC network and $\mathbf{\Theta}$ represents all trainable parameters.

2.3. Parameters setting

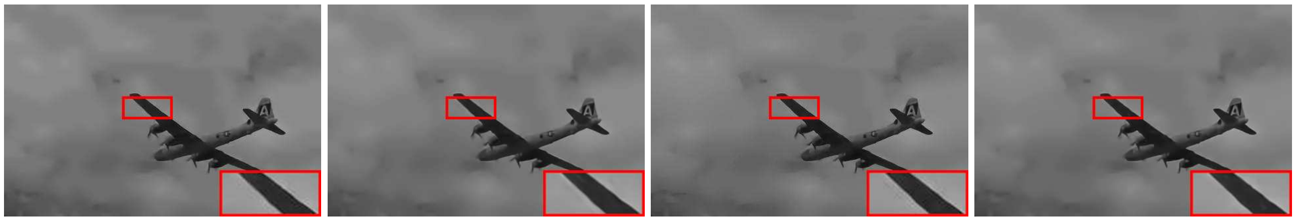
In our network architecture, all convolutional kernel sizes are 3×3 and the number of iterations R is set to 40. To keep the resolution of all the feature maps unchanged, we zero pad prior to all convolutions. The number of feature maps of each dilated convolutional layer is 32, and we set this number to 64 for the remaining convolutional layers.

2.4. Training details

We use the disjoint training set and testing set from BSD500 [3] as our training data. We use the Matlab JPEG encoder to generate JPEG compressed images by setting the input quality value to $10, 20$ and 30. We emphasize that we only train one model to handle all three JPEG qualities. The training process is conducted on the Y channel image of YCrCb space. We randomly generate one million 80×80 patch pairs for training and use TensorFlow [1] to implement our end-to-end DCSC network. We use the Adam solver [25] with a mini-batch size of 10 and fix the learning rate to 10^{-4}.

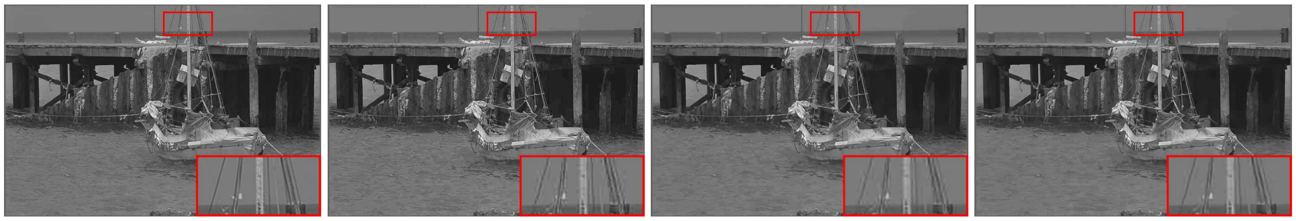


(a) GT: PSNR|SSIM|PSNR-B (b) JPEG: 34.69|0.921|34.66 (c) LD: 36.69|0.951|36.87 (d) ARCNN: 37.52|0.966|37.45

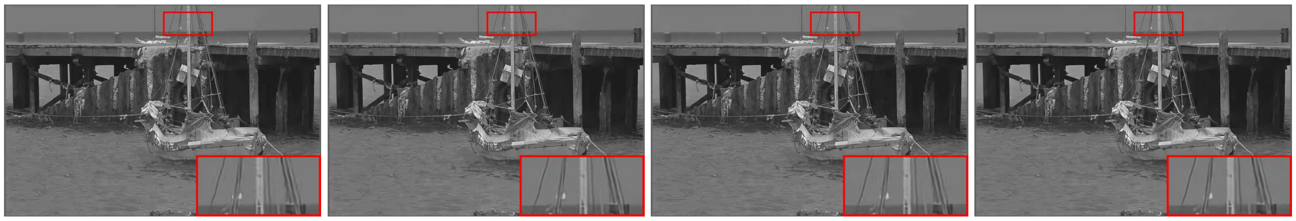


(e) TNRD: 37.81|0.968|37.74 (f) DnCNN: **38.19**|0.970|38.11 (g) LPIO: 38.10|0.970|38.02 (h) Ours: 38.17|**0.971**|**38.14**

Figure 5. Visual comparison on a JPEG compressed image (quality = 10) from the BSD500 dataset. Please zoom in for better visualization.



(a) GT: PSNR|SSIM|PSNR-B (b) JPEG: 30.47|0.827|30.47 (c) LD : 31.25|0.836|31.14 (d) ARCNN: 31.57|0.847|31.54



(e) TNRD: 31.68|0.850|31.62 (f) DnCNN: **31.75**|0.853|31.67 (g) LPIO: 31.33|0.853|31.33 (h) Ours: 31.72|**0.853**|**31.70**

Figure 6. Visual comparison on a JPEG compressed image (quality = 20) from the LIVE1 dataset. Please zoom in for better visualization.

3. Experiments

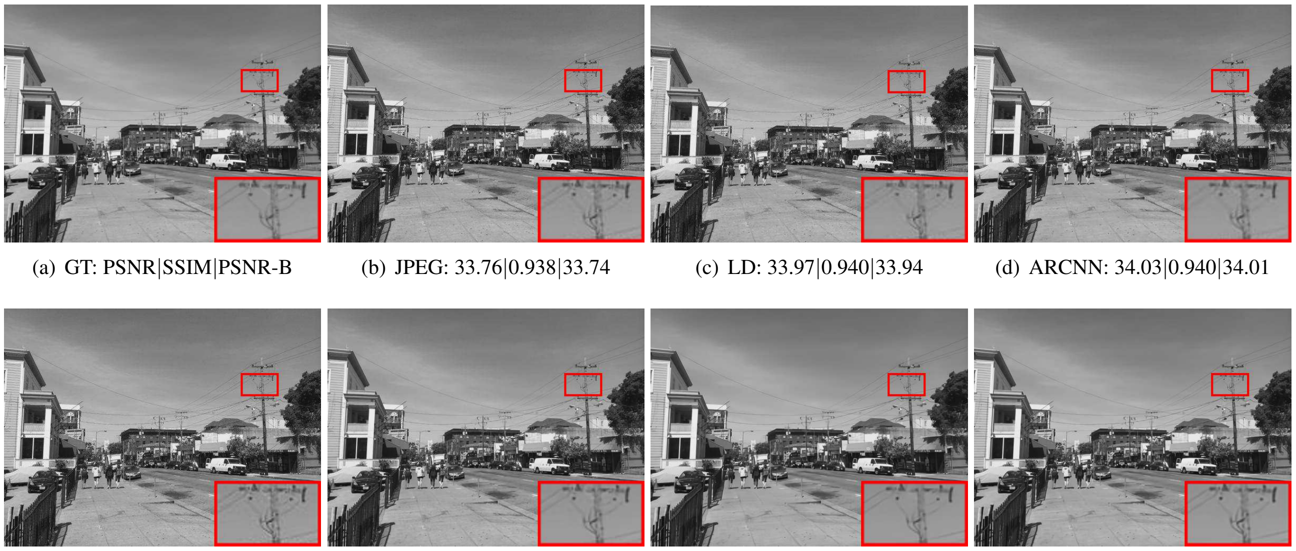
3.1. Comparison with state-of-the-art methods

We compare our network with one model-based method, Layer Decomposition-based (LD) [29], and four learning-based methods, Artifacts Reduction Convolutional Neural Network (ARCNN) [8], Trainable Nonlinear Reaction Diffusion (TNRD) [6], Denoising Convolutional Neural Network (DnCNN) [53] and Learning Parameterized Image Operators (LPIO) [10]. For testing we adopt the Classic5 [48] (5 images), LIVE1 [38] (29 images) and the validation set of BSD500 [3] (100 images) as data sets. Figures 5 and 6 show visual results of two JPEG compressed images with quality = 10 and 20, respectively. As can be seen, our DCSC has comparable visual results with DnCNN and LPI-O while outperforming other methods. Moreover, DnCNN and LPIO contain slight artifacts around the edges, as shown in the red rectangles. In terms of computation, to process a 1024×1024 image, our DCSC needs 1.4 seconds on CPU and 0.3 seconds on GPU. All experiment were run on a PC with 2 Intel i7-8700 CPUs and 1 GTX 1080Ti GPU.

We also calculate the PSNR, structural similarity (SSIM) [40], and PSNR-B [44] for quantitative assessment. PSNR-B is recommended [8] for use in this problem since it is designed to be more sensitive to blocking artifacts than SSIM. The quantitative results are shown in Table 1. Our method has comparable PSNR and SSIM values with DnCNN [53] and the best PSNR-B results on all JPEG qualities. The

Table 1. PSNR|SSIM|PSNR-B values and parameter numbers comparisons. The best and the second best results are boldfaced and underlined. Note that our DCSC achieves promising results on PSNR and SSIM and best results on PSNR-B, which is specifically designed for evaluating this de-blocking task, with relative less parameter numbers.

Dataset	Quality	LD [29]	ARCNN [8]	TNRD [6]	DnCNN [53]	LPIO [10]	Our DCSC
Classic5	10	28.39|0.800|27.59	29.03|0.793|28.78	29.28|0.799|29.04	**29.40**|**0.803**|29.10	29.35|0.801|29.04	29.25| 0.803|**29.24**
	20	30.30|0.858|29.98	31.15|0.852|30.60	31.47|0.858|31.05	**31.63**|**0.861**|31.19	31.58|0.856|31.12	31.43| 0.860|**31.41**
	30	31.50|0.882|31.31	32.51|0.881|32.00	32.78|0.884|32.24	**32.91**|**0.886**|32.36	32.86|0.883|32.28	32.68|0.885|**32.66**
LIVE1	10	28.26|0.805|27.68	28.96|0.808|28.77	29.15|0.811|28.88	**29.19**|0.812|28.91	29.17|0.811|28.89	29.17|**0.815**|**29.17**
	20	30.19|0.871|30.08	31.29|0.873|30.79	31.46|0.877|31.04	**31.59**|**0.880**|31.08	31.52|0.876|31.07	31.48|0.880|**31.47**
	30	31.32|0.898|31.27	32.67|0.904|32.22	32.84|0.906|32.28	32.98|**0.909**| 32.35	32.99|0.907|32.31	32.83|0.909|**32.81**
BSD500	10	28.03|0.782|27.29	28.56|0.783|28.54	28.42|0.781|28.30	**28.84**|0.783|28.44	28.81|0.781|28.39	28.81|**0.784**|**28.79**
	20	29.82|0.851|29.57	30.42|0.852|30.39	30.35|0.854|30.16	**31.05**|**0.857**|30.29	30.92|0.855|30.07	30.96|0.857|**30.92**
	30	30.89|0.883|30.83	31.51|0.884|31.47	31.36|0.887|31.12	**32.36**|**0.891**|31.43	32.31|0.886|31.27	32.24|0.890|**32.19**
# Params ($\times 10^5$)		—	1.06	0.21	6.69	13.94	0.94



(a) GT: PSNR|SSIM|PSNR-B (b) JPEG: 33.76|0.938|33.74 (c) LD: 33.97|0.940|33.94 (d) ARCNN: 34.03|0.940|34.01

(e) TNRD: 34.76|0.948|34.73 (f) DnCNN: **36.53**|**0.965**|36.41 (g) LPIO: 36.12|0.962|36.08 (h) Ours: 36.51|**0.965**|**36.49**

Figure 7. Visual comparison on a JPEG compressed image from the *Twitter* dataset [8].

PSNR-B results indicate our model is more suitable for this JPEG artifacts reduction task. Moreover, compared with DnCNN, our model achieves comparable results while the parameter number is reduced by 86.49%. This is because our DCSC is derived from the classical LISTA, which improves the explainability. This explainability can guide us to improve performance by better designing the network architecture, rather than by simply stacking network layers.

3.2. Use case on *Twitter*

To further demonstrate the generalization ability of our DCSC model for real use cases, we make comparisons on the *Twitter* dataset provided by ARCNN [8]. This dataset contains 114 high-quality images and their *Twitter*-compressed versions. For this task we do not retrain any of the deep learning-based methods. Figure 7 shows one example, where we see that our model generates a comparable overall visual quality with DnCNN [53] and LPIO [10], while the edges and structures, shown in the red rectangles, are sharper than other methods in our result. Table 2 shows quantitative evaluation, where we again see our model consistently generates the best PSNR-B values. This indicates that our model has a good generalization ability and potential values for practical applications.

3.3. Ablation study

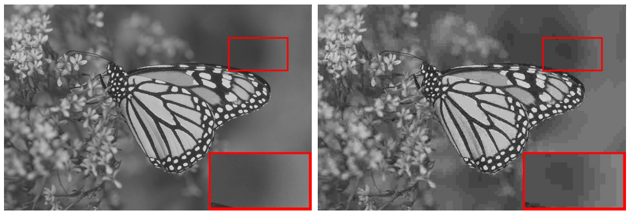
We next consider different model configurations to study their impact on performance.

3.3.1 Multi-scale features

We first assess our multi-scale features extraction strategy by training another network with the same structure, but without using dilated convolutions. Figure 8 shows one visual comparison on a JPEG compressed image with quality

Table 2. PSNR|SSIM|PSNR-B comparisons on the *Twitter* dataset [8].

LD [29]	ARCNN [8]	TNRD [6]	DnCNN [53]	LPIO [10]	Our DCSC
24.18|0.693|24.17	28.12|0.752|28.11	28.46|0.761|28.43	**28.65**|**0.770**|28.43	25.84|0.758|25.80	28.55|**0.770**|**28.54**


(a) Ground truth (b) JPEG compressed image

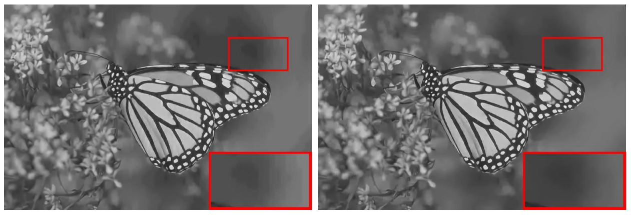

(c) w/o dilated convolution (d) w/ dilated convolution

Figure 8. Effect of dilated convolution on a JPEG compressed image with quality = 10.

Table 3. PSNR|SSIM|PSNR-B comparisons on LIVE1 dataset by using dilated convolution.

Quality	w/o dilated convolution	w/ dilated convolution
10	29.10|0.812|29.08	**29.17**|**0.815**|**29.17**
20	31.41|0.878|31.39	**31.48**|**0.880**|**31.47**
30	32.79|0.906|32.78	**32.83**|**0.909**|**32.81**

= 10. As can be seen in Figure 8(a), without using dilated convolution the de-blocked image retains obvious blocking artifacts, since multi-scale information is not modeled. This problem can be solved by stacking more convolution layers [39] to increase the receptive field at the expense of more parameters and memory requirements. As shown in Figure 8(b), using dilated convolutions can significantly reduce the blocking artifacts without increasing parameter number. A quantitative comparison on the LIVE1 dataset is also shown in Table 3, where we see that using multi-scale features improves the result.

3.3.2 Number of filters and iterations

Intuitively, the performance can be improved by increasing the network in two dimensions, either the number of filters or the depth (or in our case, iterations). We test the impact of these two factors on the LIVE1 dataset with quality = 10. Specifically, we test for depth $R \in \{20, 40, 60\}$ and filter numbers $\in \{32, 64, 128\}$. As shown in Table 4, adding more iterations achieves more obvious improvement, in agreement with in LISTA [16]. Increasing the number of filters can improve the linear representation of the network, which has limited help in solving nonlinear learning problems. To balance the trade-off between effectiveness and efficiency, we choose $R = 40$ and filter number $= 64$ as a default setting.

Table 4. PSNR|SSIM|PSNR-B comparisons on LIVE1 dataset by using different numbers of filters and iterations (quality = 10).

	filters # =32	filters # =64	filters # =128
$R = 20$	28.34|0.801|28.32	28.46|0.806|28.41	28.67|0.809|28.64
$R = 40$	29.14|0.813|29.10	29.17|0.815|29.17	29.19|0.815|29.17
$R = 60$	29.19|0.814|29.16	29.21|0.815|29.18	29.25|0.817|29.23
Params #	0.38×10^5	0.94×10^5	2.60×10^5

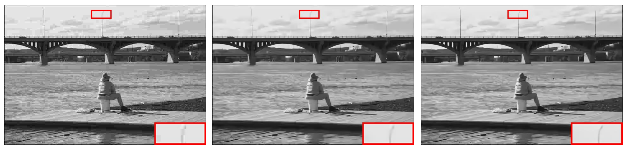

(a) JPEG image (b) MSE loss (c) MAE loss

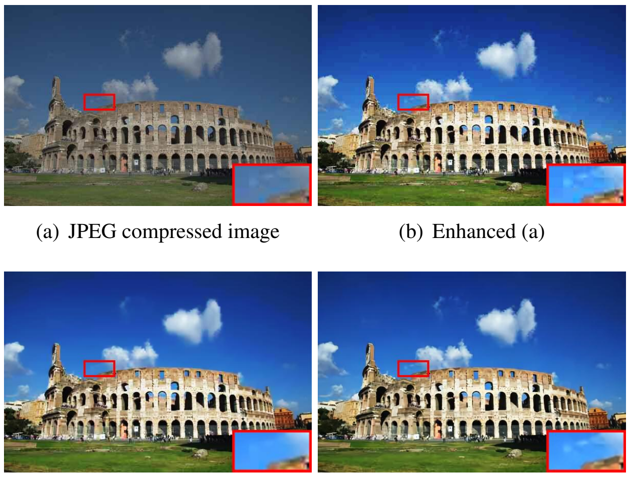
Figure 9. Comparison on different loss functions. Using MAE loss generates a sharper result.

Table 5. PSNR|SSIM|PSNR-B comparisons on LIVE1 dataset using different losses.

Quality	DnCNN [53]	Ours (MSE loss)	Ours (MAE loss)
10	**29.19**|0.812|28.91	29.13| 0.806|29.15	29.17|**0.815**|**29.17**
20	**31.59**|**0.880**|31.08	31.34| 0.876|31.33	31.48|0.880|**31.47**
30	32.98|**0.909**| 32.35	32.70| 0.905|32.41	32.83|0.909|**32.81**

3.3.3 Loss function

We use the MAE loss since it does not over-penalize larger errors and thus can preserves structure and edges. On the contrary, the widely used MSE loss, on which PSNR is based, often generates over-smoothed results because it penalizes larger errors and tolerates small errors. Therefore, MSE struggles to preserve image structures compared to MAE. Figure 9 shows two results generated by MSE and MAE, respectively. As can be seen, using MAE can preserve more details. A quantitative comparison on LIVE1 dataset is also conducted and shown in Table 5. Note that our model achieves better PSNR-B results when both our method and DnCNN use MSE loss.



(a) JPEG compressed image (b) Enhanced (a)

(c) LD [29] (d) (b) + our post-processing

Figure 10. Post-processing for image enhancement. Our model simultaneously achieves artifacts removal and content preservation.

3.4. Applications

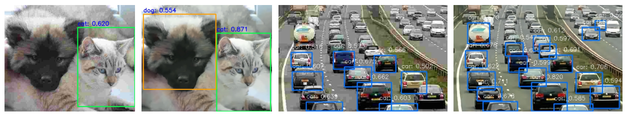
Our DCSC model can also be trained on color images and applied to other vision tasks, which we discuss below.

3.4.1 Post-processing for image enhancement

Image enhancement is useful for edge detection, object segmentation and many other vision tasks. However, enhancement algorithms usually boost not only image appearance but also JPEG artifacts, which affects both performance and perception. In this case, we found that applying our method as a post-processing is useful. In Figure 10, we show an example by comparing with LD [29], which is designed for joint image enhancement and artifacts reduction. As can be seen, using our DCSC improves the visual quality of an enhanced JPEG image. Moreover, compared with LD, our model can preserve sharper edges and more content, *e.g.*, the cloud, shown in the red rectangles.

3.4.2 Pre-processing for high-level vision tasks

Most existing models for high-level vision tasks are trained using high quality images. These learned models will have degraded performance when applied to JPEG compressed images, even if the problem is no more difficult to the human eye. In this case, a JPEG artifacts reduction model can be useful for these high-level vision applications. To test whether using our model can improve the detection performance, we adopt the YOLO [37] algorithm on JPEG-compressed images. Figure 11 shows two visual results in which the dog and some cars are not detected in the JPEG compressed images. On the contrary, using our DCSC as pre-processing the detection is improved by detecting the dog and more cars, having higher confidence scores, and having more accurate positions of the bounding box.



(a) JPEG (b) Our result (c) JPEG (d) Our result

Figure 11. Pre-processing for object detection (threshold = 0.5).

4. Analysis

Our network architecture is constructed by following Equations (1), (3), (5) and (6), and is therefore compact. Moreover, our DCSC has obvious differences with other related methods. For example, in [41], the authors build a dual-domain network, which contains two modules for the DCT and pixel domains, by combining model-based and learning-based methods. However, this model only uses a one-step shallow SC inference for each module and directly maps JPEG images to de-blocked images. In contrast, our approach iteratively performs CSC inference on multi-scale features. Compared with [41], our deep model is able to capture greater contextual information.

Interestingly, as shown in Figure 1, adding $(\mathbf{G} * \mathbf{X})$ can be seen as an identity connection, which coincides with the network structure of ResNet [21]. Adding $(\mathbf{G} * \mathbf{X})$ represents the data fidelity term while using the identity connection of ResNet aims to train very deep network. For deep models, both can effectively propagate information during the feed-forward process and address gradient vanishing during back-propagation. Meanwhile, If the input $\mathbf{X}$ is sequential or changed during inference, our model becomes the standard RNN form. In other words, RNN can be seen as a special case of the classical LISTA with sequential inputs. This may provide new ideas for exploring the internal link between model-based methods and deep learning.

5. Conclusion

In this work, based on a combination of convolutional sparse coding and deep learning, we design a explainable network to reduce JPEG artifacts from single image. We also use dilated convolution to allow our single model to handle artifacts at different scales resulting from different JPEG compression levels. The network architecture is simple, small in scale, and we believe intuitive, while still achieving competitive de-blocking performance. Finally, our DCSC approach has potential value for other vision tasks. In future work, we will explore integration of JPEG-related penalties, *e.g.*, in both image and DCT domains, into our model.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *Symposium on Operating Systems Design and Implementation*, 2016. 4

[2] Michal Aharon, Michael Elad, Alfred Bruckstein, et al. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006. 3

[3] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(5):898–916, 2011. 4, 5

[4] Hilton Bristow, Anders Eriksson, and Simon Lucey. Fast convolutional sparse coding. In *CVPR*, 2013. 1, 3

[5] Huibin Chang, Michael K Ng, and Tieyong Zeng. Reducing artifacts in JPEG decompression via a learned dictionary. *IEEE Transactions on Signal Processing*, 62(3):718–728, 2014. 2, 3

[6] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, 2017. 1, 2, 5, 6, 7

[7] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007. 2

[8] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. Compression artifacts reduction by a deep convolutional network. In *ICCV*, 2015. 1, 2, 4, 5, 6, 7

[9] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014. 2

[10] Qingnan Fan, Dongdong Chen, Lu Yuan, Gang Hua, Nenghai Yu, and Baoquan Chen. Decouple learning for parameterized image operators. In *ECCV*, 2018. 2, 5, 6, 7

[11] Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images. *IEEE Transactions on Image Processing*, 16(5):1395–1411, 2007. 1, 2

[12] Xueyang Fu, Jiabin Huang, Delu Zeng, Yue Huang, Xinghao Ding, and John Paisley. Removing rain from single images via a deep detail network. In *CVPR*, 2017. 4

[13] Xueyang Fu, Delu Zeng, Yue Huang, Xiao-Ping Zhang, and Xinghao Ding. A weighted variational model for simultaneous reflectance and illumination estimation. In *CVPR*, 2016. 3

[14] Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. Deep generative adversarial compression artifact removal. In *ICCV*, 2017. 1, 2

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 2

[16] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *ICML*, 2010. 1, 2, 3, 7

[17] Shuhang Gu, Deyu Meng, Wangmeng Zuo, and Lei Zhang. Joint convolutional analysis and synthesis sparse representation for single image layer separation. In *ICCV*, 2017. 1, 3

[18] Shuhang Gu, Wangmeng Zuo, Qi Xie, Deyu Meng, Xiangchu Feng, and Lei Zhang. Convolutional sparse coding for image super-resolution. In *ICCV*, 2015. 1

[19] Jun Guo and Hongyang Chao. Building dual-domain representations for compression artifacts reduction. In *ECCV*, 2016. 2

[20] Jun Guo and Hongyang Chao. One-to-many network for visually pleasing compression artifacts reduction. In *CVPR*, 2017. 2

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 8

[22] Felix Heide, Wolfgang Heidrich, and Gordon Wetzstein. Fast and flexible convolutional sparse coding. In *CVPR*, 2015. 1

[23] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 2

[24] Jeremy Jancsary, Sebastian Nowozin, and Carsten Rother. Loss-specific training of non-parametric image restoration models: A new state of the art. In *ECCV*, 2012. 2

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 4

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 3

[27] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 2

[28] Tao Li, Xiaohai He, Linbo Qing, Qizhi Teng, and Honggang Chen. An iterative framework of cascaded deblocking and superresolution for compressed images. *IEEE Transactions on Multimedia*, 20(6):1305–1320, 2018. 2

[29] Yu Li, Fangfang Guo, Robby T Tan, and Michael S Brown. A contrast enhancement framework with JPEG artifacts suppression. In *ECCV*, 2014. 1, 2, 5, 6, 7, 8

[30] AW-C Liew and Hong Yan. Blocking artifacts suppression in block-coded images using overcomplete wavelet representation. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(4):450–461, 2004. 2

[31] Peter List, Anthony Joch, Jani Lainema, Gisle Bjontegaard, and Marta Karczewicz. Adaptive deblocking filter. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):614–619, 2003. 1, 2

[32] Xianming Liu, Gene Cheung, Xiaolin Wu, and Debin Zhao. Random walk graph laplacian-based smoothness prior for soft decoding of JPEG images. *IEEE Transactions on Image Processing*, 26(2):509–524, 2017. 2

[33] Xianming Liu, Xiaolin Wu, Jiantao Zhou, and Debin Zhao. Data driven sparsity based restoration of JPEG compressed images in dual transform pixel domain. In *CVPR*, 2015. 1, 2

[34] Xianming Liu, Xiaolin Wu, Jiantao Zhou, and Debin Zhao. Data-driven soft decoding of compressed images in dual transform-pixel domain. *IEEE Transactions on Image Processing*, 25(4):1649–1659, 2016. 2

[35] James G Nagy and Dianne P O'Leary. Restoring images degraded by spatially variant blur. *SIAM Journal on Scientific Computing*, 19(4):1063–1082, 1998. 3

[36] Vardan Papyan, Yaniv Romano, Jeremias Sulam, and Michael Elad. Convolutional dictionary learning via local processing. In *ICCV*, 2017. 1

[37] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 8

[38] HR Sheikh. LIVE image quality assessment database release 2. 2005. 5

[39] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *ICCV*, 2017. 2, 7

[40] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 5

[41] Zhangyang Wang, Ding Liu, Shiyu Chang, Qing Ling, Yingzhen Yang, and Thomas S Huang. D^3: Deep dual-domain based fast restoration of JPEG-compressed images. In *CVPR*, 2016. 1, 2, 8

[42] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010. 3

[43] Yongyi Yang, Nikolas P Galatsanos, and Aggelos K Katsaggelos. Projection-based spatially adaptive reconstruction of block-transform compressed images. *IEEE Transactions on Image Processing*, 4(7):896–908, 1995. 2

[44] Changhoon Yim and Alan Conrad Bovik. Quality assessment of deblocked images. *IEEE Transactions on Image Processing*, 20(1):88–98, 2011. 5

[45] Jaeyoung Yoo, Sang-ho Lee, and Nojun Kwak. Image restoration by estimating frequency distribution of local patches. In *CVPR*, 2018. 2

[46] Seok Bong Yoo, Kyuha Choi, and Jong Beom Ra. Post-processing for blocking artifact reduction based on inter-block correlation. *IEEE Transactions on Multimedia*, 16(6):1536–1548, 2014. 1, 2

[47] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 2

[48] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *International Conference on Curves and Surfaces*, 2010. 5

[49] Guangtao Zhai, Wenjun Zhang, Xiaokang Yang, Weisi Lin, and Yi Xu. Efficient deblocking with coefficient regularization, shape-adaptive filtering, and quantization constraint. *IEEE Transactions on Multimedia*, 10(5):735–745, 2008. 2

[50] He Zhang and Vishal Patel. Convolutional sparse coding-based image decomposition. In *BMVC*, 2016. 1

[51] He Zhang and Vishal M Patel. Convolutional sparse and low-rank coding-based image decomposition. *IEEE Transactions on Image Processing*, 27(5):2121–2133, 2018. 1, 3

[52] Jian Zhang, Ruiqin Xiong, Chen Zhao, Yongbing Zhang, Siwei Ma, and Wen Gao. CONCOLOR: Constrained non-convex low-rank model for image deblocking. *IEEE Transactions on Image Processing*, 25(3):1246–1259, 2016. 2

[53] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017. 1, 2, 4, 5, 6, 7

[54] Xinfeng Zhang, Weisi Lin, Ruiqin Xiong, Xianming Liu, Siwei Ma, and Wen Gao. Low-rank decomposition-based restoration of compressed images via adaptive noise estimation. *IEEE Transactions on Image processing*, 25(9):4158–4171, 2016. 2

[55] Xiaoshuai Zhang, Wenhan Yang, Yueyu Hu, and Jiaying Liu. DMCNN: Dual-domain multi-scale convolutional neural network for compression artifacts removal. In *ICIP*, 2018. 1