A Model-Driven Deep Unfolding Method for JPEG Artifacts Removal

Xueyang Fu, Menglu Wang, Xiangyong Cao, Xinghao Ding, and Zheng-Jun Zha, *Member, IEEE*

***Abstract*— Deep learning-based methods have achieved notable progress in removing blocking artifacts caused by lossy JPEG compression on images. However, most deep learning-based methods handle this task by designing black-box network architectures to directly learn the relationships between the compressed images and their clean versions. These network architectures are always lack of sufficient interpretability, which limits their further improvements in deblocking performance. To address this issue, in this article, we propose a model-driven deep unfolding method for JPEG artifacts removal, with interpretable network structures. First, we build a maximum posterior (MAP) model for deblocking using convolutional dictionary learning and design an iterative optimization algorithm using proximal operators. Second, we unfold this iterative algorithm into a learnable deep network structure, where each module corresponds to a specific operation of the iterative algorithm. In this way, our network inherits the benefits of both the powerful model ability of data-driven deep learning method and the interpretability of traditional model-driven method. By training the proposed network in an end-to-end manner, all learnable modules can be automatically explored to well characterize the representations of both JPEG artifacts and image content. Experiments on synthetic and real-world datasets show that our method is able to generate competitive or even better deblocking results, compared with state-of-the-art methods both quantitatively and qualitatively.**

***Index Terms*— Convolutional dictionary, deep learning, image restoration, JPEG artifacts removal, optimization.**

I. Introduction

With the rapid development of imaging equipment and social media, high-resolution images and videos are exploding and spreading every day. To overcome the bandwidth-hungry bottleneck caused by the visual data explosion, lossy compression technologies, such as JPEG [1] and high-efficiency video coding (HEVC) [2], have been widely used in various imaging devices and softwares. However, due to the signal removal in the compression process, compressed images and videos usually contain visually unpleasing artifacts, e.g., blocking, blurring, and banding effects. These compression artifacts severely deteriorate not only the quality in visual perception but also feature fidelity in computer vision tasks. Therefore, removing these artifacts from compressed images and videos is an important postprocessing task and has drawn much research attention in recent years [3]–[6]. In this article, we focus on removing artifacts generated by the JPEG compression, which is the most common compression strategy.

Based on the redundancies in spatial and transform domains, JPEG compression first divides the image into nonoverlapping 8×8 pixel blocks and then applies a discrete cosine transformation (DCT) to convert each block into transformed coefficients. By utilizing predefined quantization steps, these DCT coefficients are further coarsely quantized to remove high-frequency parts of the image. The reason for this quantization scheme is because the human vision system is less sensitive to these high-frequency parts, such as object textures. The compressed image is finally obtained by applying an inverse DCT to the quantized coefficients. Due to discontinuity at the boundaries and removal of high-frequency parts of the image, obvious blocking and blurring artifacts will appear in the compressed image. In addition, a larger quantization step can save more storage space, but at the cost of banding effects.

To reduce undesirable JPEG compression artifacts and improve the images quality, many methods have been proposed. In general, JPEG artifacts removal methods can be classified into two categories: model-driven methods and data-driven deep learning. The model-driven methods are usually designed by filtering JPEG artifacts or treating this task as an ill-posed inverse problem. For instance, based on the shape-adaptive DCT (SA-DCT), Foi *et al.* [7] proposed a filtering method for image noise removal and compression artifacts reduction. Yoo *et al.* [8] utilized the interblock correlation to remove blocking artifacts in smooth regions. On the other hand, based on the maximum post probability framework, many researchers aim to extract clean images via solving an optimization objective function. Since multiple latent clean versions can be estimated from a single compressed image, the inherently ill-posed property in this task requires prior knowledge to provide effective regularization. Along this research direction, many prior models, e.g., sparse representation [9]–[11], low rank [12], [13], nonlocal self-similarity [14], and graph [15], [16], have been developed to form the regularization terms and constrain the solution space. Due to the handcrafted prior assumptions, the representation abilities of these model-driven methods are limited, which leads to

Manuscript received May 15, 2020; revised December 6, 2020 and April 8, 2021; accepted May 20, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2020AAA0105702; in part by the National Natural Science Foundation of China (NSFC) under Grant 61901433, Grant 61906151, Grant U19B2038, and Grant 61620106009; in part by the University Synergy Innovation Program of Anhui Province under Grant GXXT-2019-025; and in part by the USTC Research Funds of the Double First-Class Initiative under Grant YD2100002003. *(Corresponding author: Xiangyong Cao.)*

Xueyang Fu, Menglu Wang, and Zheng-Jun Zha are with the School of Information Science and Technology, University of Science and Technology of China, Hefei 230026, China.

Xiangyong Cao is with the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: caoxiangyong@mail.xjtu.edu.cn).

Xinghao Ding is with the School of Informatics, Xiamen University, Xiamen 361005, China.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3083504.

Digital Object Identifier 10.1109/TNNLS.2021.3083504

unstable results when processing compressed images with complex structures. In addition, these methods often require time-consuming iterative calculations, which greatly reduce efficiency.

Inspired by the breakthrough of data-driven deep learning in various vision tasks [17]–[22], many methods based on deep convolutional neural networks (CNNs) have been proposed for JPEG artifacts removal. The current DL-based methods mainly focus on training a mapping relationship, i.e., a deep network, from compressed images to the desired clean version on abundant compressed/clean image pairs. Typical deblocking network structures include shallow models [3], deep models [23], multiscale U-Net [24], and so on. However, one obvious deficiency of these deep learning-based methods is the lack of interpretability. Since most network modules are empirically designed, the entire architectures can be seen as black-box mechanisms, and thus, the role of different modules in these networks is difficult to analyze and understand.

To address the aforementioned issues, a hybrid deblocking method that combines the model-based sparse coding and deep networks has been proposed in [25]. Compared with other sparse coding-based methods, the integration of deep models can extract rich representations from large training datasets and thus leads to superior deblocking performance. Along this research line, in this article, we propose a new convolutional model-driven deep unfolding network for JPEG artifacts removal. We aim to design an interpretable deep network, which combines advantages of both the convolutional model-driven optimization and data-driven deep learning, for this specific image deblocking task. Overall, our contributions are mainly threefold.

1) We propose a convolutional dictionary model to encode JPEG artifacts for image deblocking. To effectively solve this model, we introduce an optimization algorithm, whose iterative process can be efficiently carried out, based on proximal gradient techniques. This optimization algorithm is directly performed in image domain without complex transformations, e.g., Fourier transformation and DCT. This helps the algorithm to be easily unfolded into a deep structure by common network modules.
2) We propose a deep network architecture by unfolding the iterative algorithm for image deblocking. Since each network module corresponds to a specific iterative step, the feedforward process of the entire network mimics the signal process flow of the optimization algorithm, which increases the interpretability of the deep model. Moreover, we incorporate multiscale dilated convolutions into the deep network to make it better remove wide-range distortions and handle multiple JPEG factors.
3) By training our model in an end-to-end manner, all learnable parameters can be automatically explored to well estimate both JPEG artifacts and clean image content. Extensive experiments show that our model not only has increased interpretability but also achieves notable improvements over state-of-the-art methods both quantitatively and qualitatively.

A preliminary conference version of this work was presented earlier [6]. The present work adds to the initial version in significant ways, and in the following, we summarize the changes. First, compared to our previous method that only considers the image layer, we add the JPEG artifacts layer into our convolutional model to make it more comprehensive. Second, instead of using the fixed ℓ_1 sparse regularization terms in the previous model, we relax the regularizer forms and automatically learn them from training data. Third, our previous network architecture is derived from the learned iterative shrinkage threshold algorithm (LISTA) [26] and performed in the feature domain. In this work, we utilize the proximal gradient technique to construct the deep network and extend it in the image domain. Fourth, we provide more comprehensive evaluations and add considerable analyses to demonstrate the effectiveness of our model.

II. Related Work

In this section, we briefly review the most related artifacts removal methods for JPEG compressed images, i.e., model-driven methods and data-driven deep learning-based methods

A. Model-Driven Methods

In early image deblocking methods [27], [28], filtering operations are explicitly designed and performed to remove compression artifacts. For instance, Minami and Zakhor [29] proposed an adaptive method for estimating the parameters of filters. A quadratic programming problem is introduced to achieve blocking artifacts removal and details preservation. Foi *et al.* [7] proposed a shape-adaptive DCT-based filtering operation, which is able to achieve both the image denoising and deblocking. Norkin *et al.* [30] introduced an in-loop filter to remove the blocking artifacts between coding units. A nonlocal means filtering operation is well designed in [31] to reduce blocking artifacts. Based on the correlation of interblock, Yoo *et al.* [8] adopted different strategies to reduce blocking artifacts in smooth regions and edge regions.

Instead of designing explicit filtering operations, some researchers treat JPEG artifacts removal as an ill-posed inverse problem and solve it in the framework of maximum posterior (MAP) probability estimation. Hence, to effectively constraining solution space, complex image prior is explored and utilized, such as nonlocal similarity [14], low-rank minimization [12], [13], [32], sparse representation [9]–[11], and graph [15], [16]. Sun and Cham [33] used the additive Gaussian noise and Markov random field to express the distortion and the image, respectively. Zhang *et al.* [14] removed compression artifacts by exploring the nonlocal similarity in DCT domain. To simultaneously utilize both the nonlocal similarity and local sparsity, the method [32] combines patch clustering and low-rank minimization. Li *et al.* [34] adopted the image decomposition technique and sparse prior to achieve both JPEG artifacts removal and image enhancement. Based on the Retinex theory, Liu *et al.* [16] designed a graph smoothness prior to jointly remove compression artifacts and improve illumination contrast. Although these model-driven methods are flexible and have good interpretability, they

usually require time-consuming optimizations and all variables cannot be jointly optimized.

B. Data-Driven Deep Learning Methods

In the past decade, due to the powerful representation learning ability, deep learning has made breakthrough progress in various high-level vision tasks [17]–[19]. Inspired by these great successes, deep CNNs have also been studied to deal with low-level image processing [22], [35]–[37]. For JPEG artifacts reduction, Dong *et al.* [3] proposed the first deep learning-based method by designing a four-layer CNN architecture. Inspired by this seminal work, some methods aim to explore more effective network architecture to improve the deblocking performance. Based on the encoder–decoder architecture, symmetric convolutional–deconvolutional networks [38], [39] are proposed to extract multiscale features for compression artifact suppression. Zhang *et al.* [23] adopted the residual learning to construct a deep network for image denoising and deblocking. To fully strengthen the long-term dependence of deep features, Tai *et al.* [40] proposed a deep persistent memory network (MemNet) to adaptively fuse the feature maps at different layers. Fan *et al.* [41] proposed a decouple learning framework to incorporate different parameterized image operators for image filtering and restoration tasks. To utilize long-range dependencies information, Zhang *et al.* [42] introduced a residual nonlocal network by taking nonlocal block [43] into consideration. To generate more realistic textures and details, Galteri *et al.* [44], [45] proposed a generative adversarial network to capture the underlying data distribution for compression artifacts reduction. A similar strategy is also proposed in [4] to produce visually pleasing results. Zhang *et al.* [46] proposed a residual dense block to extract multilevel features from different layers. To further facilitate JPEG artifacts removal, some dual-domain-based deep networks are proposed. Wang *et al.* [25] utilized domain knowledge and first incorporated the DCT-domain prior into deep networks. Guo and Chao [47] constructed a dual-domain network in both DCT domain and pixel domain to suppress compression artifacts. Zhang *et al.* [48] integrated the dual-domain and dilated convolutions to eliminate both blocking and banding artifacts. To achieve a good balance between the receptive fields and computational efficiency, wavelet transforms are introduced in [24]. Chen *et al.* [49] introduced a two-branch CNN in both image domain and wavelet domain to handle the JPEG compression artifacts. Zheng *et al.* [5] proposed an extractor–corrector framework by implicitly utilizing the DCT-domain information. Recently, an inception-based deep network [50] is introduced to perform both blind and nonblind compression artifact removal. To effectively recover high-frequency details, Yoo *et al.* [51] formulated deblocking as a classification in the DCT domain.

At present, some works attempt to leverage both the domain knowledge and deep networks for image processing. For instance, to obtain speed and performance gains, Wang *et al.* [25] built a cascaded network to perform the LISTA [26] in dual domain. Chen and Pock [52] implemented the classic iterative nonlinear reaction–diffusion method as a deep network for fast and effective image restoration. Yang *et al.* [53] combined traditional compressive sensing and deep networks for image reconstruction using sparsely sampled measurements. Li *et al.* [54] unrolled the iterative algorithm to construct a deep network for blind image deblurring. Under the alternative minimization framework, deep CNNs are further utilized to learn regularizers [21], [55]. Albeit obtaining initial success, there are also some handcrafted operations in the above methods. Meanwhile, deep CNNs are still adopted as the main backbone, which thus still lack sufficient interpretability. Instead of learning regularizers, deep CNNs have also been used as proximal operators used in optimization algorithms in pixel domain [56]. Our network architecture shares these similar spirits, but unlike the above methods, we utilize convolution dictionary learning to model JPEG artifacts to better handle this specific task.

III. Methodology

A. Preliminaries on Deep Convolutional Sparse Coding [6]

In the preliminary conference version [6], the deblocking process is expressed as

$$\mathbf{O} = f_{\mathrm{DCSC}}(\mathbf{J}) \tag{1}$$

where $f_{\mathrm{DCSC}}(\cdot)$ is the deep convolutional sparse coding (DCSC) network and $\mathbf{J}$ and $\mathbf{O}$ denote the JPEG compressed image and the desired clean version, respectively. To combine the merits of model-driven methods and deep learning, the classic CSC is adopted to form the objective in the feature domain. Then, the LISTA [26] is adopted to solve the objective. Finally, the DCSC network architecture is constructed by following the iteration steps of LISTA. Note that the DCSC is only performed in the feature domain and directly predicts the deblocked image. In other words, the prior knowledge of both image content and JPEG artifacts is not fully explored, which limits the interpretability and performance of the deep model. Therefore, it is possible to explore more reasonable objective functions and optimization algorithms to guide network design and further improve deblocking performance.

B. Objective Formulation

Before describing our deep unfolding model in detail, we first formulate the objective for JPEG artifacts reduction. Given a gray-level JPEG compressed image $\mathbf{J}$, it can be rationally modeled as

$$\mathbf{J} = \mathbf{O} + \mathbf{H} \tag{2}$$

where $\mathbf{O}$ and $\mathbf{H}$ represent the desired clean image and JPEG artifacts. It is clear that (2) has inherently ill-posed property since numerous $\mathbf{O}$ and $\mathbf{H}$ can be generated from a single $\mathbf{J}$. To handle this ill-posed inverse problem, most deep learning-based methods empirically design complex deep architectures to learn the mapping function from $\mathbf{J}$ to $\mathbf{O}$. Hence, the image prior is implicitly embedded into the deep network and explored from training data. To explicitly and fully utilize domain knowledge, we first explore the prior

knowledge for representing JPEG artifacts $\mathbf{H}$. Specifically, we adopt a convolutional model to express $\mathbf{H}$ as

$$\mathbf{H} = \sum_{n=1}^{N} W_n \otimes M_n \tag{3}$$

where $\{W_n\}_n$ is a set of convolutional kernels to describe the local patterns of JPEG artifacts, $\{M_n\}_n$ is the corresponding coefficients to represent local patterns, N is the number of kernels, and $\otimes$ is the 2-D convolutional operation. Therefore, (2) can be rewritten as

$$\mathbf{J} = \mathbf{O} + \sum_{n=1}^{N} W_n \otimes M_n. \tag{4}$$

Note that (3) can be seen as the form of classic convolutional dictionary [57], [58], which aims to represent repetitive local spatial patterns. The reasons we choose convolutional dictionary to model artifacts $\mathbf{H}$ are threefold. First, since the convolutional operation is spatially invariant [58] and can directly process the entire image, it is suitable for handling low-level image processing tasks. Second, compared to complex image content in $\mathbf{O}$, the spatial patterns in $\mathbf{H}$ are relatively simple. Since the patterns of JPEG artifacts, e.g., blocking and banding, usually appear repeatedly, these artifacts are very suitable to be expressed using a convolutional dictionary. Third, adopting the convolutional model in the image domain can not only avoid complex operations, e.g., Fourier transform and DCT, but also can be directly used for subsequent deep CNNs design. Since these convolutional dictionaries represent common knowledge shared by different JPEG artifacts, they can be learned from training data with the help of powerful representation capabilities of deep learning.

To effectively estimate both $\mathbf{O}$ and $\mathbf{H}$ from $\mathbf{J}$, based on the maximum *a posteriori* framework, the solution can be obtained by maximizing the joint probability $P(\mathbf{O}, \mathbf{H}|\mathbf{J})$

$$\arg\max_{\mathbf{O},\mathbf{H}} P(\mathbf{O}, \mathbf{H}|\mathbf{J}) = \arg\max_{\mathbf{O},\mathbf{H}} P(\mathbf{J}|\mathbf{O}, \mathbf{H}) P(\mathbf{O}) P(\mathbf{H}) \tag{5}$$

where $P(\mathbf{J}|\mathbf{O}, \mathbf{H})$ denotes the conditional probability of the JPEG compressed image $\mathbf{J}$ and $P(\mathbf{O})$ and $P(\mathbf{H})$ are the prior knowledge of $\mathbf{O}$ and $\mathbf{H}$, respectively. Specifically, we assume the residual error $\mathbf{E} = \mathbf{J} - \mathbf{O} - \mathbf{H}$ to obey a multivariate Gaussian distribution $N(\mathbf{E}|\mathbf{0}, \sigma^2\mathbf{I})$, and thus, the observed image $\mathbf{J}$ follows the following Gaussian distribution: $\mathbf{J}|\mathbf{O}, \mathbf{H} \sim N(\mathbf{J}|\mathbf{O} + \mathbf{H}, \sigma^2\mathbf{I})$. Besides, we assume that $\mathbf{O}$ and $\mathbf{H}$ obey the following distributions, namely $P(\mathbf{H}) = (1/C_1)e^{(-\lambda_1 f_1(\mathbf{H}))}$ and $P(\mathbf{O}) = (1/C_2)e^{(-\lambda_2 f_2(\mathbf{O}))}$, where C_1 and C_2 are normalization terms and λ_1 and λ_2 are distribution parameters. By performing a negative logarithmic transformation, (5) can be rewritten as an energy minimization model

$$\arg\min_{\mathbf{O},\mathbf{H}} \|\mathbf{J} - \mathbf{O} - \mathbf{H}\|_F^2 + \lambda_1 f_1(\mathbf{H}) + \lambda_2 f_2(\mathbf{O}) \tag{6}$$

where $f_1(\mathbf{H})$ and $f_2(\mathbf{O})$ denote the regularizers associated with the prior terms $P(\mathbf{H})$ and $P(\mathbf{O})$, respectively. Inspired by the classic CSC model [57], [58] that explores prior on the coefficients, the final objective derived from (6) and (4) can be modeled as

$$\arg\min_{\mathbf{O},\mathbf{M}} \left\| \mathbf{J} - \mathbf{O} - \sum_{n=1}^{N} W_n \otimes M_n \right\|_F^2 + \lambda_1 f_1(\mathbf{M}) + \lambda_2 f_2(\mathbf{O}) \tag{7}$$

where $\mathbf{M}$ denotes the tensor form of all M_n.

C. Optimization

In general, the above objective function (7) can be handled by alternatively solving two subproblems

$$\mathbf{M}^{(t)} = \arg\min_{\mathbf{M}} \left\| \mathbf{J} - \mathbf{O}^{(t-1)} - \sum_{n=1}^{N} W_n \otimes M_n \right\|_F^2 + \lambda_1 f_1(\mathbf{M}) \tag{8}$$

$$\mathbf{O}^{(t)} = \arg\min_{\mathbf{O}} \left\| \mathbf{J} - \mathbf{O} - \sum_{n=1}^{N} W_n \otimes M_n^{(t)} \right\|_F^2 + \lambda_2 f_2(\mathbf{O}) \tag{9}$$

where t is the current stage. However, conventional optimization methods [58], [59] for convolutional problems require complicated operations, e.g., fast Fourier transform (FFT) and inverse FFT, which are not very friendly for deep network design. Therefore, inspired by the proximal gradient method [60], we introduce a simple yet effective optimization algorithm to iteratively calculate $\mathbf{M}$ and $\mathbf{O}$. In the following, we detail the updates for each subproblem.

1) M-Subproblem: Instead of directly optimizing the objective (8), we calculate the coefficients $\mathbf{M}$ based on the quadratic approximation of (8) [60], which is expressed as

$$\mathbf{M}^{(t)} = \arg\min_{\mathbf{M}} g\left(\mathbf{M}^{(t-1)}\right) + \frac{1}{2\eta_1}\left\|\mathbf{M} - \mathbf{M}^{(t-1)}\right\|_F^2 + \left\langle \mathbf{M} - \mathbf{M}^{(t-1)}, \nabla g\left(\mathbf{M}^{(t-1)}\right)\right\rangle + \lambda_1 f_1(\mathbf{M}) \tag{10}$$

where $g(\mathbf{M}^{(t-1)}) = \|\mathbf{J} - \mathbf{O}^{(t-1)} - \sum_{n=1}^{N} W_n \otimes M_n^{(t-1)}\|_F^2$, $\nabla(\cdot)$ denotes the gradient operator, and η_1 is the step size. Note that the problem (10) is equivalent to

$$\mathbf{M}^{(t)} = \arg\min_{\mathbf{M}} \frac{1}{2}\left\|\mathbf{M} - \left(\mathbf{M}^{(t-1)} - \eta_1 \nabla g\left(\mathbf{M}^{(t-1)}\right)\right)\right\|_F^2 + \eta_1 \lambda_1 f_1(\mathbf{M}). \tag{11}$$

The solution of (11) can be obtained by utilizing the form of general regularization terms [60] and is defined as

$$\mathbf{M}^{(t)} = \text{prox}_{\eta_1 \lambda_1}\left(\mathbf{M}^{(t-1)} - \eta_1 \nabla g\left(\mathbf{M}^{(t-1)}\right)\right). \tag{12}$$

Note that $\nabla g(\mathbf{M}^{(t-1)})$ is expressed as

$$\nabla g\left(\mathbf{M}^{(t-1)}\right) = \mathbf{W} \otimes^{\dagger} \left(\sum_{n=1}^{N} W_n \otimes M_n^{(t-1)} + \mathbf{O}^{(t-1)} - \mathbf{J} \right) \tag{13}$$

where $\mathbf{W}$ is a 4-D tensor stacked by all W_n and $\otimes^{\dagger}$ denotes the transposed convolution. Finally, by bringing (13) into (12), the updated $\mathbf{M}^{(t)}$ can be expressed as

$$\mathbf{M}^{(t)} = \text{prox}_{\eta_1 \lambda_1}\left(\mathbf{M}^{(t-1)} - \eta_1 W \otimes^{\dagger} \left(\sum_{n=1}^{N} W_n \otimes M_n^{(t-1)} + \mathbf{O}^{(t-1)} - \mathbf{J} \right) \right) \tag{14}$$

where $\text{prox}_{\eta_1 \lambda_1}(\cdot)$ is the proximal operator that related to the regularization term $f_1(\cdot)$. To avoid the limitation

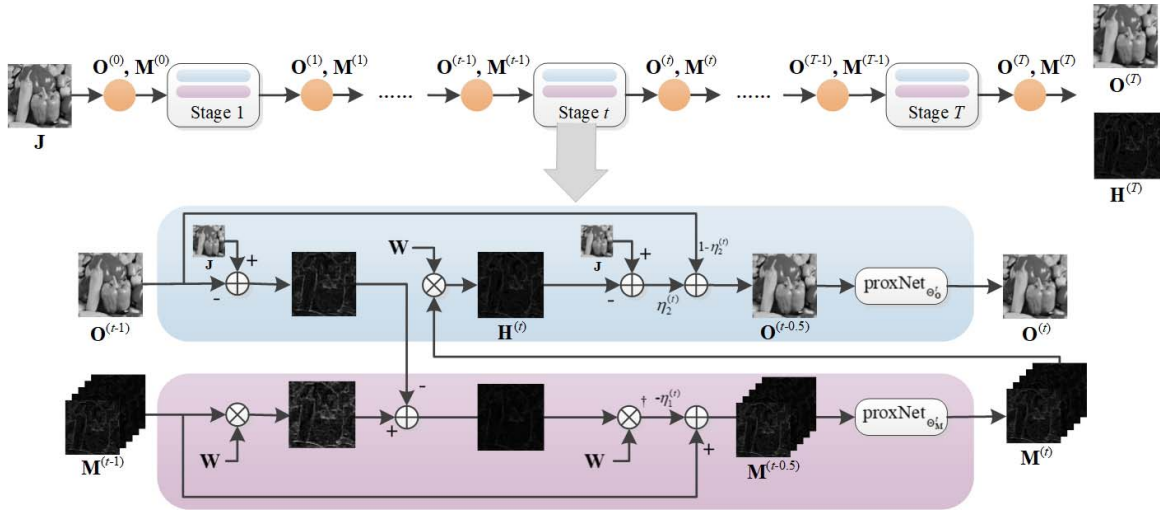


Fig. 1. Framework of our convolutional model-driven deep unfolding with T stages. The overall optimization process and specific operations are shown on the top row and bottom row, respectively. The network takes a JPEG compressed image $\mathbf{J}$ as input and simultaneously outputs the compression artifacts $\mathbf{H}$ and deblocked image $\mathbf{O}$. $\otimes$ and $\oplus$ denote the convolutional operation and elementwise addition, respectively.

of human-designed regularizers, we utilize deep CNNs to automatically explore prior, which will be detailed in Section III-D.

2) O-Subproblem: We adopt a similar strategy to update $\mathbf{O}$. First, the quadratic approximation of the problem (9) with respect to $\mathbf{J}$ can be expressed as

$$\mathbf{O}^{(t)} = \arg\min_{\mathbf{O}} h\left(\mathbf{O}^{(t-1)}\right) + \frac{1}{2\eta_2}\left\|\mathbf{O} - \mathbf{O}^{(t-1)}\right\|_F^2 + \left\langle \mathbf{O} - \mathbf{O}^{(t-1)}, \nabla h\left(\mathbf{O}^{(t-1)}\right)\right\rangle + \lambda_2 f_2(\mathbf{O}) \quad (15)$$

where $h(\mathbf{O}^{(t-1)}) = \|\mathbf{J} - \mathbf{O}^{(t-1)} - \sum_{n=1}^{N} W_n \otimes M_n^{(t)}\|_F^2$ and η_2 is the step size. Similarly, the problem (15) is equivalent to

$$\mathbf{O}^{(t)} = \arg\min_{\mathbf{O}} \frac{1}{2}\left\|\mathbf{O} - \left(\mathbf{O}^{(t-1)} - \eta_2 \nabla h\left(\mathbf{O}^{(t-1)}\right)\right)\right\|_F^2 + \eta_2\lambda_2 f_2(\mathbf{O}) \quad (16)$$

where $\nabla h(\mathbf{O}^{(t-1)})$ is expressed as

$$\nabla h\left(\mathbf{O}^{(t-1)}\right) = \sum_{n=1}^{N} W_n \otimes M_n^{(t)} + \mathbf{O}^{(t-1)} - \mathbf{J}. \quad (17)$$

Finally, the updated $\mathbf{O}^{(t)}$ can be expressed as

$$\mathbf{O}^{(t)} = \mathrm{prox}_{\eta_2\lambda_2}\left(\mathbf{O}^{(t-1)} - \eta_2\mathbf{O}^{(t-1)} + \eta_2\left(\mathbf{J} - \sum_{n=1}^{N} W_n \otimes M_n^{(t)}\right)\right) \quad (18)$$

where $\mathrm{prox}_{\eta_2\lambda_2}(\cdot)$ is the proximal operator that related to the regularization term $f_2(\cdot)$.

D. Deep Unfolding Network Architecture

To gain advantages from both model- and data-driven methods [21], [22], [55], [56], [61], we unfold the above optimization algorithm into deep networks for JPEG artifacts removal. Each network module is intentionally designed to correspond to each step in the optimization algorithm. In this way, not only the design of prior is simplified but also the interpretability of the deep model is improved. Our proposed unfolding network architecture is shown in Fig. 1.

As shown in (14) and (18), the two proximal operators $\mathrm{prox}_{\eta_1\lambda_1}(\cdot)$ and $\mathrm{prox}_{\eta_2\lambda_2}(\cdot)$ play a key role of the optimization algorithm. To update each variable, conventional methods usually take visual cues to design model priors and then use them to represent the two proximal operators. However, due to the limited representation abilities, human-designed priors cannot always achieve the expected effect. Therefore, different from previous works [6], [62] that use manually designed priors, we utilize deep networks to explore prior for both $\mathbf{M}$ and $\mathbf{O}$. In other words, for (14) and (18), we do not design explicit model for the two proximal operators. In this way, the explicit description and design of complex priors can be avoided.

Corresponding to (14) and (18), our deep unfolding network is designed into several stages, and each stage updates $\mathbf{M}$ and $\mathbf{O}$ in a cascaded manner. Specially, at stage t, $\mathbf{M}^{(t)}$ is first updated by taking the JPEG compressed image $\mathbf{J}$, the previous $\mathbf{M}^{(t-1)}$ and $\mathbf{O}^{(t-1)}$ as inputs

$$\mathbf{M}^{(t-0.5)} = \mathbf{M}^{(t-1)} - \eta_1 \mathbf{W} \otimes^{\dagger} \left(\sum_{n=1}^{N} W_n \otimes M_n^{(t-1)} - (\mathbf{J} - \mathbf{O}^{(t-1)})\right) \quad (19)$$

$$\mathbf{M}^{(t)} = \mathrm{proxNet}_{\Theta_{\mathbf{M}}^{(t)}}\left(\mathbf{M}^{(t-0.5)}\right). \quad (20)$$

Then, $\mathbf{O}^{(t)}$ is further updated by using $\mathbf{J}$, the previous $\mathbf{O}^{(t-1)}$ and current $\mathbf{M}^{(t)}$

$$\mathbf{H}^{(t)} = \sum_{n=1}^{N} W_n \otimes M_n^{(t)} \quad (21)$$

$$\mathbf{O}^{(t-0.5)} = \mathbf{O}^{(t-1)} - \eta_2\mathbf{O}^{(t-1)} + \eta_2\left(\mathbf{J} - \mathbf{H}^{(t)}\right) \quad (22)$$

$$\mathbf{O}^{(t)} = \mathrm{proxNet}_{\Theta_{\mathbf{O}}^{(t)}}\left(\mathbf{O}^{(t-0.5)}\right) \quad (23)$$

where $\mathrm{proxNet}_{\Theta_{\mathbf{M}}^{(t)}}(\cdot)$ and $\mathrm{proxNet}_{\Theta_{\mathbf{O}}^{(t)}}(\cdot)$ are two residual networks (ResNets) [18], respectively. $\Theta_{\mathbf{M}}^{(t)}$ and $\Theta_{\mathbf{O}}^{(t)}$ are the parameters of ResNets. Note that all the network parameters

can be end-to-end learned from training data. Once the training is finished, the testing time will be shortened due to its feed-forward architecture. We initialize $\mathbf{M}^{(0)}$ as $\mathbf{0}$ and obtain $\mathbf{O}^{(0)}$ by simply using one standard 3×3 convolution operation: $\mathbf{O}^{(0)} = \mathbf{W}^{(0)} \otimes \mathbf{J} + \mathbf{b}^{(0)}$.

E. Interpretability

As shown in Fig. 1, our deep unfolding network has a good interpretability. Specifically, the residual information of the coefficients, which is obtained by $\mathbf{W}\otimes^{\dagger}(\sum_{n=1}^{N} W_n \otimes M_n^{(t-1)} - (\mathbf{J} - \mathbf{O}^{(t-1)}))$ in (19), is actually the gradient that represents the increasing direction. Hence, the network modules related to $\mathbf{M}^{(t-0.5)}$ represents the classic gradient descent process, while the module of $\text{proxNet}_{\Theta_{\mathbf{M}}^{(t)}}(\cdot)$ not only extracts priors but also performs nonlinear operations to further update $\mathbf{M}^{(t-0.5)}$. This is similar to the classic LISTA method [26], which adopts the ℓ_1-norm to perform sparse prior and uses the soft-threshold operation as the proximal operator. The network modules related to $\mathbf{O}$ have the same interpretation. Therefore, each network module has a clear meaning, i.e., corresponds to each step in the optimization algorithm.

F. Network Design and Loss Function

To achieve a good balance between the performance and the cost of training, we utilize the classic but capable ResNets [18] to perform the proximal operators for updating $\mathbf{M}$ and $\mathbf{O}$. The reasons we choose ResNets are threefold. First, as one of the most popular deep networks, the shortcut connections in ResNets allow the construction of very deep networks, which makes the proposed residual network more powerful nonlinear approximation ability to the proximal operators. Second, the residual structure has been widely used in many image restoration tasks [42], which proves that ResNets are also suitable for low-level image processing in addition to high-level vision tasks. Last but not least, compared with other more advanced and complex deep networks, the structure of residual networks is relatively simple, which can better isolate the benefit of our framework itself. Specifically, we design the deep CNNs, which contain ten residual blocks, to implicitly explore priors. The residual block consists of two convolutional layers, each followed by a nonlinear activation. Moreover, we adopt dilated convolution [63] to perform $\mathbf{W}$ for multiscale features extraction. By dilating the same filter to different scales, dilated convolutions can enlarge the receptive field without introducing extra parameters. This helps to remove wide-range distortions and enables a single network to handle multiple JPEG compression factors.

For image restoration tasks, the most widely used loss function is mean squared error (mse) [23], [46]. However, due to the squared penalty that works poorly at image details and edges, mse usually generates oversmoothed results. Instead, we use the mean absolute error (MAE) to train our deep unfolding network. MAE does not overpenalize larger errors and thus can preserve details and edges, which is similar with the total variation minimization applications. Given L training image pairs $\{\mathbf{J}_l, \mathbf{O}_{l,gt}\}_{l=1}^{L}$, we minimize the following objective function at every stage:

$$\frac{1}{L}\sum_{l=0}^{L}\left(\sum_{t=0}^{T}\alpha^{(t)}\left\|\mathbf{O}_l^{(t)}-\mathbf{O}_{l,gt}\right\|_1+\sum_{t=1}^{T}\beta^{(t)}\left\|\mathbf{H}_l^{(t)}-\left(\mathbf{J}_l-\mathbf{O}_{l,gt}\right)\right\|_1\right) \tag{24}$$

where gt denotes the ground truth, $\mathbf{O}^{(t)}$ and $\mathbf{H}^{(t)}$ denote the deblocked image and JPEG artifacts at stage t, respectively, and $\alpha^{(t)}$ and $\beta^{(t)}$ are the tradeoff parameter for stage t. In all experiments, we set $\alpha^{(t)} = \beta^{(t)} = 0.1$ $(t = 0, 2, \ldots, T-1)$ and $\alpha^{(T)} = \beta^{(T)} = 1.0$, to make the final stage play a dominant role, while other parameters are used for stable training.

G. Difference Between Our Method and DCSC [6]

The main difference between our method and DCSC [6] lies in three aspects. First, for the design of the objective function, this work incorporates both image layer $\mathbf{O}$ and JPEG artifacts layer $\mathbf{H}$ into the fidelity term, which makes it more comprehensive. Meanwhile, different from the predefined ℓ_1-norm used in [6] as the regularization term, we relax the constraints and adopt deep ResNets to explore prior knowledge from training data. Compared with the relatively simple ℓ_1 sparse prior, the use of deep networks as the proximal operators can further increase the model capacity and make the regularization term more general. Second, for the design of the deep unfolding algorithm, DCSC directly builds the network structure based on LISTA [26] at the feature level, while this work designs the network structure based on the proximal gradient algorithm [60] at both image and feature levels, which makes the network more comprehensive. For each variable to be solved in the objective function, we build a corresponding network module to mimic its solving process in the optimization algorithm. Therefore, the interpretability of this work is better than DCSC. Third, for the design of the loss function, DCSC only uses a single loss in the final stage, while this work performs multistage supervision losses on the intermediate results. Since our network mimics the iterative process of the optimization algorithm, the use of multistage losses facilitates the deep network to obtain a better updating direction for each iterative stage.

H. Implementation Details

In our deep unfolding architecture, all convolutional kernel sizes are set as 3×3. The number of feature maps of each convolutional layer is 48. The number of stages T is set to 10. To keep the resolution of all the feature maps unchanged, we zero pad prior to all convolutional operations. We use the classic ReLU [17] as the nonlinear activation. The dilation factors are set as 1, 2, and 4. Note that our proposed optimization algorithm, which is used to solve the variational model, only contains simple operators. The network developed based on the iterative algorithm can be easily constructed by general network modules, such as the standard 2-D convolution, ReLU activation, and elementwise additions. According to the experimental results, our unfolding network that contains these simple operators can already achieve promising deblocking performance. Moreover, the use of simple operators is conducive to the analysis of the function

of each network module. In addition, existing open-source software libraries, e.g., TensorFlow or PyTorch, can easily implement our proposed network architecture. This allows our network to be easily modified for other image processing tasks, e.g., super-resolution (SR) shown in the extension.

To train the network, we use the MATLAB JPEG encoder to generate JPEG compressed images. The JPEG quality factors (QFs) are set to 10, 20, 30, and 40, and we use both the training and testing sets from BSD500 [64] as our training set. The training process is conducted on the Y channel image of YCrCb space. We randomly generate 64×64 training patch pairs and use TensorFlow [65] to implement our deep unfolding network. The Adam solver [66] with a mini-batch size of 10 is used as the optimizer. The learning rate is fixed to 10^{-4}. We only train one single model to handle all the four JPEG compression factors.

IV. Experiments

To demonstrate the superior performance of our deep unfolding for JPEG artifact removal, both the quantitative and qualitative evaluations on synthetic datasets and real-world use case are conducted. We compare our model with state-of-the-art methods, including two model-driven methods, SA-DCT [7] and layer decomposition (LD) [34], and several deep learning-based methods, including artifacts reduction CNN (ARCNN) [3], trainable nonlinear reaction diffusion (TNRD) [52], denoising CNN (DnCNN) [23], learning parameterized image operators (LPIOs) [41], MemNet [40], knowledge-driven layer separation (KDLS) [21], DCSC [6], inception-based artifact removal CNN (IACNN) [50], and residual nonlocal attention networks (RNANs) [42].

A. Comparisons on Synthetic Datasets

We first report the quantitative assessment results of different methods on the three synthetic datasets, i.e., Classic5 [67] (five images), LIVE1 [68] (29 images), and the validation set of BSD500 [64] (100 images). We adopt the peak signal-to-noise ratio (PSNR), structural similarity (SSIM) [69], and peak signal-to-noise ratio including blocking effects (PSNR-B) [70] for quantitative evaluations. Since PSNR-B is more sensitive to blocking artifacts than SSIM, it is recommended [3] for use in this deblocking problem.

1) Quantitative Comparisons: The quantitative comparisons on grayscale images are shown in Table I. It is clear that our deep unfolding network achieves the best overall performances on all datasets at all JPEG QFs. Generally, RNAN [42] generates the second best results in PSNR, SSIM, and PSNR-B. In particular, for the grayscale Classic5 dataset in the case of QF $= 10$, the average gains of our method over MemNet and DCSC are, respectively, 0.26 and 0.33 dB in PSNR, 0.0236 and 0.0073 in SSIM, and 0.30 and 0.31 dB in PSNR-B. When compared with the other methods, our method is far ahead. In particular, as shown in Table I at QF $= 10$, the average improvements of our deep unfolding networks over the popular ARCNN [3], DnCNN [23] and KDLS [21] are, respectively, up to 0.92, 0.55, and 0.74 dB in PSNR, 0.0414, 0.0317, and 0.0073 in SSIM, and 0.85, 0.48, and 0.83 dB in PSNR-B. Although the recently proposed RNAN method and our method have close PSNR values, a significant improvement can be observed in the comparison of SSIM and PSNR-B, which indicates that our model achieves better structure restoration and blocking artifacts removal.

2) Qualitative Comparisons: Our deep unfolding model not only outperforms all the above comparative methods in terms of quantitative evaluations but also generates appealing visual results among them. Figs. 2 and 3 show two deblocked images generated by various image approaches at QF $= 10$ and 20, respectively. It can be seen from Fig. 2 that other compared methods can effectively remove most blocking artifacts, but they fail to well recover object details, as shown in the enlarged region. On the contrary, RNAN and our model can restore the stripe patterns with better visual quality. Moreover, our model achieves better edge protection than RNAN, as shown between the leg and arm. In Fig. 3, the enlarged regions contain another kind of compression artifact, i.e., blurring artifacts. It is clear that the results generated by LD [34], ARCNN [3], and KDLS [21] suffer from obvious blurring artifacts around edges. MemNet [40] and DCSC [6] can alleviate it to some degree, and our model can further reduce blurring artifacts and recover clearer edges. We also show color-scale deblocking visual results in Figs. 4 and 5. The enlarged regions contain both compression artifacts with color shift. For instance, in Fig. 4, LD [34], ARCNN [3], and KDLS [21] tend to generate obvious artifacts along the edges. TNRD [52] is unable to recover clear edges. In Fig. 5, MemNet [40], IACNN [50], and RNAN [42] could generate blurring artifacts along edges. In contrast, our model can simultaneously handle compression artifacts and recover the clean smooth area.

B. Generalization Ability

As a popular online social media, Twitter is widely used for message publishing and sharing. However, to reduce storage and transmission consumption, this platform compresses and rescales the original uploaded images on the server side. Therefore, when other users view the posted image, they will observe complex compression artifacts. We adopt the Twitter dataset [3], which contains 114 compression/clean image pairs, and compare our model with other methods on this real-world use case. To avoid the problem of out-of-memory caused by excessive image resolution, we first crop the image and then perform the deblocking operation and measurement calculation. Fig. 6 shows visual comparisons, and it is clear that the Twitter compressed image contains different types of artifacts compared with the above synthetic images. It is clear that our model can generate clearer and sharper restoration results than other compared methods. In Table II, we further show quantitative comparisons, in which we observe that our model consistently generates the best overall performance.

To further demonstrate the generalization ability of our model, we also make comparisons on other two JPEG qualities, i.e., QF $= 5$ and 50, on the BSD500 dataset. It is worth noting that all the deep learning-based methods are not retrained on these two JPEG qualities. Table III shows the comparison results and our model consistently achieves

TABLE I

QUANTITATIVE COMPARISONS ON GRAYSCALE DATASETS. THE BEST AND THE SECOND BEST RESULTS ARE BOLDFACED AND UNDERLINED

QF	Methods	Classic5			LIVE1			BSD500		
		PSNR	SSIM	PSNR-B	PSNR	SSIM	PSNR-B	PSNR	SSIM	PSNR-B
10	JPEG	27.82	0.7595	25.21	27.77	0.7730	25.33	27.58	0.7694	24.97
	SA-DCT [7]	28.88	0.8071	28.16	28.65	0.8093	28.01	28.23	0.7780	27.38
	LD [34]	28.39	0.7997	27.59	28.26	0.8052	27.68	28.03	0.7824	27.29
	ARCNN [3]	29.03	0.7929	28.76	28.96	0.8076	28.77	28.56	0.7907	27.87
	TNRD [52]	29.28	0.7992	29.04	29.14	0.8111	28.88	28.60	0.7926	27.95
	DnCNN [23]	29.40	0.8026	29.13	29.19	0.8123	28.90	28.84	0.8006	28.44
	LPIO [41]	29.35	0.8015	29.04	29.17	0.8119	28.89	28.81	0.7815	28.39
	MemNet [40]	29.69	0.8107	29.31	29.45	0.8193	29.04	28.96	0.8039	28.56
	KDLS [21]	29.21	0.8075	28.78	29.03	0.8156	28.69	28.71	0.8006	28.26
	DCSC [6]	29.62	0.8270	29.30	29.34	0.8317	29.01	28.95	0.8050	28.55
	IACNN [50]	29.53	0.8124	29.27	28.80	0.8207	28.71	28.47	0.8031	28.36
	RNAN [42]	29.87	0.8278	29.42	29.52	0.8319	29.13	29.08	0.8054	28.48
	Ours	**29.95**	**0.8343**	**29.61**	**29.61**	**0.8370**	**29.25**	**29.18**	**0.8095**	**28.74**
20	JPEG	30.12	0.8344	27.50	30.07	0.8512	27.57	29.72	0.8519	26.97
	SA-DCT [7]	30.92	0.8663	29.75	30.81	0.8781	29.82	30.09	0.8510	28.61
	LD [34]	30.30	0.8584	29.37	30.19	0.8715	29.64	29.82	0.8514	28.43
	ARCNN [3]	31.15	0.8517	30.59	31.29	0.8733	30.79	30.43	0.8594	29.10
	TNRD [52]	31.47	0.8576	31.05	31.46	0.8769	31.04	30.51	0.8611	29.34
	DnCNN [23]	31.63	0.8610	31.19	31.59	0.8802	31.07	31.05	0.8741	30.29
	LPIO [41]	31.58	0.8567	31.12	31.52	0.8766	31.07	30.92	0.8551	30.07
	MemNet [40]	31.90	0.8658	31.29	31.83	0.8846	31.14	31.05	0.8742	30.36
	KDLS [21]	31.51	0.8604	31.07	31.32	0.8789	30.92	31.02	0.8713	30.19
	DCSC [6]	31.81	0.8804	31.34	31.70	0.8960	31.18	31.13	0.8758	30.41
	IACNN [50]	31.87	0.8729	31.18	31.76	0.8861	31.05	31.02	0.8732	30.20
	RNAN [42]	32.02	0.8790	31.26	31.87	0.8895	31.12	31.25	0.8751	30.27
	Ours	**32.11**	**0.8848**	**31.61**	**31.98**	**0.8997**	**31.42**	**31.36**	**0.8790**	**30.58**
30	JPEG	31.48	0.8666	28.94	31.40	0.8851	28.92	30.98	0.8865	28.22
	SA-DCT [7]	32.14	0.8914	30.83	32.08	0.9078	30.92	31.21	0.8838	29.34
	LD [34]	31.47	0.8830	30.17	29.41	0.8960	29.36	30.87	0.8719	29.15
	ARCNN [3]	32.51	0.8806	31.98	32.67	0.9043	32.22	31.52	0.8904	29.92
	TNRD [52]	32.78	0.8837	32.24	32.84	0.9059	32.28	31.58	0.8902	30.02
	DnCNN [23]	32.91	0.8861	32.38	32.98	0.9090	32.34	32.36	0.9049	31.43
	LPIO [41]	32.86	0.8835	32.28	32.99	0.9074	32.31	32.31	0.8866	31.27
	MemNet [40]	32.97	0.8881	32.49	33.07	0.9108	32.47	32.61	0.9072	31.15
	KDLS [21]	32.65	0.8842	32.27	32.72	0.9047	32.11	32.24	0.9028	31.14
	DCSC [6]	33.06	0.9030	32.49	33.07	0.9218	32.43	32.42	0.9057	31.52
	IACNN [50]	33.08	0.9007	32.18	33.14	0.9210	32.13	32.50	0.9056	31.42
	RNAN [42]	**33.40**	0.9014	32.35	**33.44**	0.9227	32.22	**32.70**	0.9068	31.33
	Ours	33.33	**0.9061**	**32.67**	33.38	**0.9251**	**32.65**	32.66	**0.9087**	**31.63**
40	JPEG	32.43	0.8849	29.92	32.35	0.9041	29.96	31.88	0.9057	29.13
	SA-DCT [7]	33.00	0.9055	31.59	32.99	0.9240	31.79	32.08	0.9030	29.94
	LD [34]	32.10	0.8944	30.47	31.89	0.9151	30.80	31.93	0.9021	28.89
	ARCNN [3]	32.68	0.9019	31.03	32.74	0.9196	31.25	32.23	0.9068	30.45
	TNRD [52]	32.86	0.9033	31.22	32.83	0.9206	31.28	32.27	0.9075	30.43
	DnCNN [23]	33.77	0.9141	33.23	33.96	0.9346	33.28	33.27	0.9216	32.24
	LPIO [41]	33.71	0.9134	33.19	33.87	0.9329	33.21	33.18	0.9207	32.21
	MemNet [40]	33.86	0.9164	33.21	34.27	0.9375	33.41	33.42	0.9217	32.31
	KDLS [21]	33.63	0.9127	33.15	33.79	0.9326	33.13	33.15	0.9204	32.14
	DCSC [6]	33.87	0.9153	33.30	34.02	0.9354	33.36	33.30	0.9216	32.31
	IACNN [50]	33.91	0.9141	33.04	34.06	0.9313	33.15	33.27	0.9213	32.20
	RNAN [42]	34.03	0.9167	33.19	34.27	0.9360	33.29	33.47	0.9229	32.27
	Ours	**34.11**	**0.9179**	**33.36**	**34.32**	**0.9384**	**33.51**	**33.52**	**0.9244**	**32.32**

the best overall performance. Since our deep unfolding model integrates the advantages of both model- and data-driven methods, the restoration process not only depends on common compression features learned from data but also considers the information of the current input image. Therefore, our model has a good generalization ability and is transferable to tackle practical problems.

C. Visualization

In this section, we use visualization to illustrate how the interpretability of our deep unfolding model facilitates the analysis of deep networks.

In Fig. 7, we visualize the predicted JPEG artifacts layers. It is clear that our model is able to explicitly extract proper JPEG artifacts layers. This verifies both the rationality and particularity of our design. First, as shown in Fig. 7, all artifacts layers have sparse appearances, i.e., most pixel values are equal to 0. This sparsity can make the learning process easier than directly predicting dense deblocked images. This is consistent with the idea of traditional model-driven methods, which usually utilize sparse prior, e.g., image gradients, to constrain ill-posed inverse restoration problems. Second, as the dilation factor increases, the corresponding JPEG artifacts layers contain structures at larger scales. This allows our

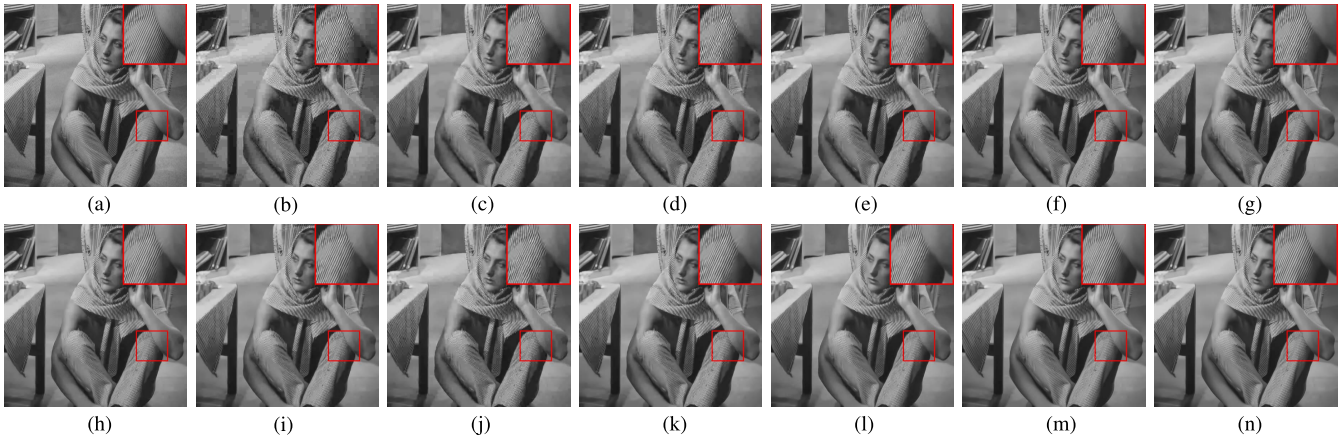


Fig. 2. Visual comparison on a grayscale JPEG compressed image (quality = 10) from the Classic5 dataset. Please zoomed-in view for better visualization on detail recovery and edge preservation. (a) Clean, PSNR | SSIM | PSNR-B. (b) JPEG, 25.79 | 0.7794 | 23.48. (c) SA-DCT [7], 26.46 | 0.8068 | 25.27. (d) LD [34], 26.18 | 0.7975 | 25.93. (e) ARCNN [3], 26.92 | 0.8120 | 26.74. (f) TNRD [52], 27.24 | 0.8247 | 27.13. (g) DnCNN [23], 27.59 | 0.8302 | 27.30. (h) LPIO [41], 26.51 | 0.8238 | 26.02. (i) MemNet [40], 28.08 | 0.8419 | 27.73. (j) KDLS [21], 27.46 | 0.8276 | 26.88. (k) DCSC [6], 28.16 | 0.8461 | 27.77. (l) IACNN [50], 27.57 | 0.8241 | 27.34. (m) RNAN [42], 28.07 | 0.8577 | 27.85. (n) Our, 28.99 | 0.8652 | 28.61.

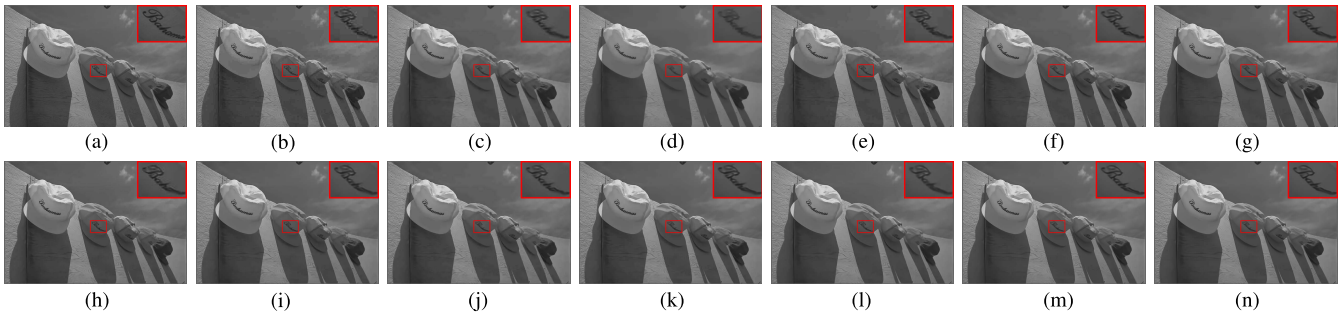


Fig. 3. Visual comparison on a grayscale JPEG compressed image (quality = 10) from the Classic5 dataset. Please zoomed-in view for better visualization on detail recovery and edge preservation. (a) Clean, PSNR | SSIM | PSNR-B. (b) JPEG, 33.97 | 0.8921 | 31.44. (c) SA-DCT [7], 34.57 | 0.9020 | 33.80. (d) LD [34], 34.05 | 0.8940 | 33.76. (e) ARCNN [3], 35.31 | 0.9121 | 34.95. (f) TNRD [52], 35.61 | 0.9178 | 35.56. (g) DnCNN [23], 35.68 | 0.9192 | 35.43. (h) LPIO [41], 32.41 | 0.9090 | 31.88. (i) MemNet [40], 35.80 | 0.9213 | 35.66. (j) KDLS [21], 35.61 | 0.9178 | 35.56. (k) DCSC [6], 35.83 | 0.9221 | 35.63. (l) IACNN [50], 35.20 | 0.9104 | 35.20. (m) RNAN [42], 35.81 | 0.9195 | 35.49. (n) Our, 36.08 | 0.9252 | 35.89.

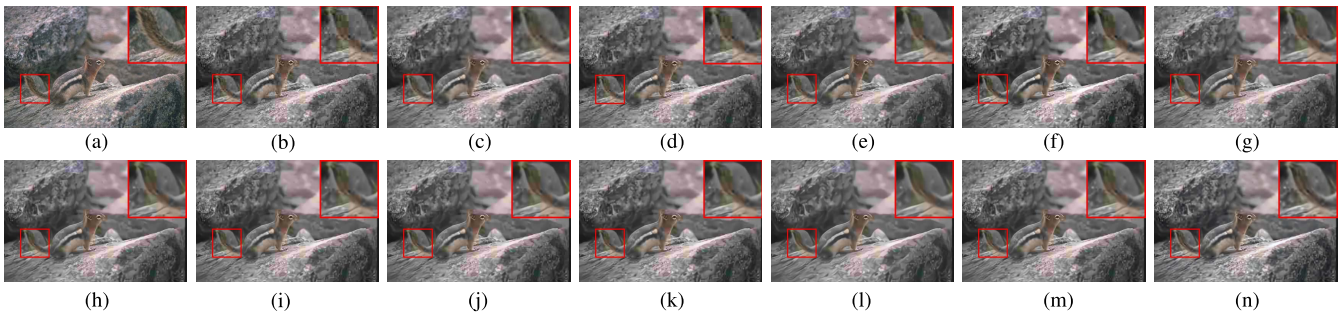


Fig. 4. Visual comparison on a color-scale JPEG compressed image (quality = 10) from the BSD500 dataset. Please zoomed-in view for better visualization on blocking artifacts removal. (a) Clean, PSNR | SSIM | PSNR-B. (b) JPEG, 29.22 | 0.7995 | 26.23. (c) SA-DCT [7], 29.98 | 0.8152 | 29.50. (d) LD [34], 29.74 | 0.8127 | 29.31. (e) ARCNN [3], 30.17 | 0.8235 | 29.44. (f) TNRD [52], 30.18 | 0.8239 | 29.61. (g) DnCNN [23], 30.40 | 0.8340 | 30.01. (h) LPIO [41], 28.79 | 0.8125 | 28.31. (i) MemNet [40], 30.45 | 0.8359 | 30.17. (j) KDLS [21], 30.27 | 0.8328 | 29.90. (k) DCSC [6], 30.46 | 0.8380 | 30.19. (l) IACNN [50], 29.92 | 0.8363 | 29.92. (m) RNAN [42], 30.30 | 0.8376 | 29.91. (n) Our, 30.55 | 0.8394 | 30.34.

model to handle not only small-scale blocking artifacts but also wide-range banding effects, as shown in Fig. 4. Third, as can be seen, all artifacts layers contain similar local patterns. This proves that it is reasonable to utilize the convolutional model and learn shared convolutional kernels to express repetitive local patterns of JPEG artifacts. This facilitates the general availability of our model to real-world compression scenarios, as demonstrated in Section IV-B.

In Fig. 8, we visualize the intermediate results $\mathbf{O}^{(t)}$ at different stages. As can be seen, our deep unfolding algorithm converges within several iterations. As the iteration progresses, $\mathbf{O}^{(t)}$ is gradually ameliorated and contains less artifacts with higher visual quality. This is consistent with traditional optimization methods, i.e., with many iterations, $\mathbf{O}$ converges to the optimal solutions. Since we can clearly observe the changing state of the intermediate $\mathbf{O}$, our deep unfolding model has good interpretability with easily visualized explanation. Note that our network module one-to-one corresponds with each iterative step in the optimization algorithm and has proper constraints from the learned prior for JPEG artifacts. Therefore, our model can be evolved to a right direction and thus obtain promising deblocking performance.

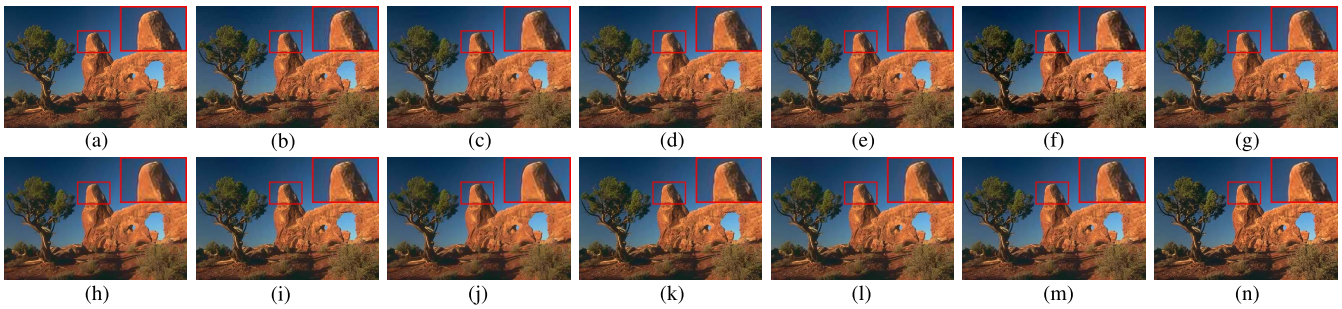


Fig. 5. Visual comparison on a color-scale JPEG compressed image (quality = 30) from the BSD500 dataset. Please zoomed-in view for better visualization on banding artifacts removal. (a) Clean, PSNR | SSIM | PSNR-B. (b) JPEG, 30.33 | 0.8755 | 27.86. (c) SA-DCT [7], 30.51 | 0.8769 | 29.11. (d) LD [34], 30.45 | 0.8752 | 29.06. (e) ARCNN [3], 30.66 | 0.8786 | 29.31. (f) TNRD [52], 30.67 | 0.8811 | 29.43. (g) DnCNN [23], 31.26 | 0.8970 | 30.37. (h) LPIO [41], 27.26 | 0.7658 | 26.83. (i) MemNet [40], 30.33 | 0.8981 | 30.06. (j) KDLS [21], 30.21 | 0.8967 | 29.89. (k) DCSC [6], 31.29 | 0.8974 | 30.51. (l) IACNN [50], 30.96 | 0.8961 | 30.93. (m) RNAN [42], 31.25 | 0.8950 | 30.32. (n) Our, 31.41 | 0.9002 | 30.49.

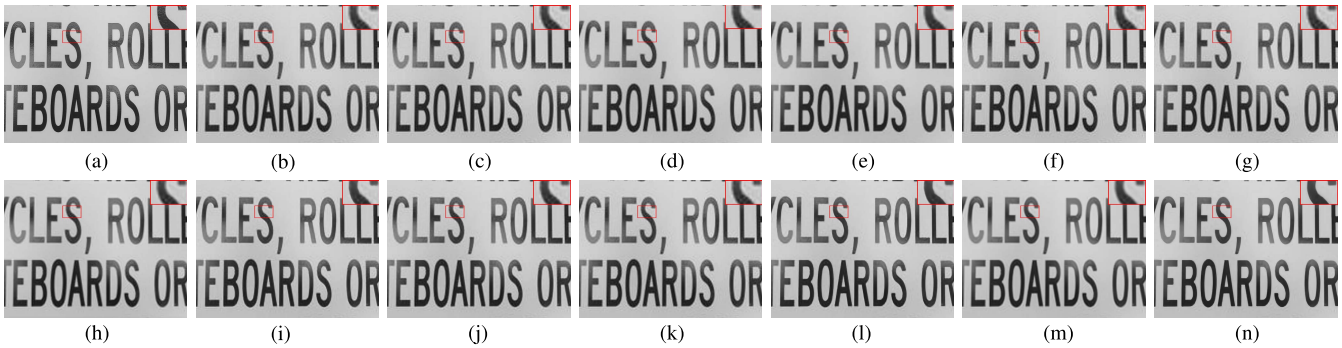


Fig. 6. Visual comparison on a compressed image from the real-world Twitter dataset. Please zoomed-in view for better visualization on compression artifacts removal. (a) Clean, PSNR | SSIM | PSNR-B. (b) JPEG, 25.17 | 0.7617 | 25.12. (c) SA-DCT [7], 25.18 | 0.7639 | 25.13. (d) LD [34], 25.06 | 0.7586 | 25.01. (e) ARCNN [3], 25.32 | 0.8001 | 25.29. (f) TNRD [52], 25.17 | 0.7615 | 25.12. (g) DnCNN [23], 25.19 | 0.7661 | 25.19. (h) LPIO [41], 23.00 | 0.7905 | 23.00. (i) MemNet [40], 27.54 | 0.8319 | 27.50. (j) KDLS [21], 26.41 | 0.8033 | 26.41. (k) DCSC [6], 25.20 | 0.7698 | 25.20. (l) IACNN [50], 28.89 | 0.8579 | 28.89. (m) RNAN [42], 27.16 | 0.8418 | 27.16. (n) Our, 30.21 | 0.8872 | 30.21.

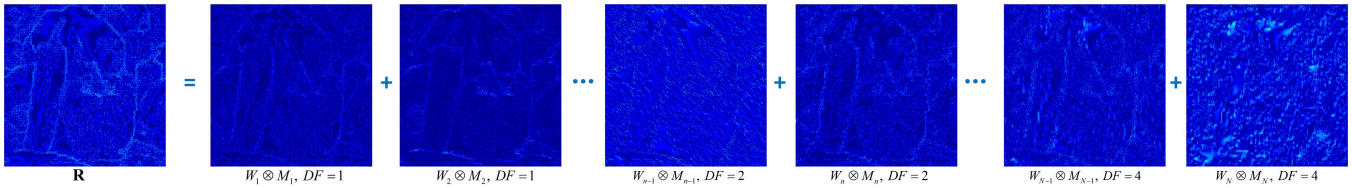


Fig. 7. Estimated compression artifacts layers at the final stage and DF denotes the dilated factor. As DF increases, the corresponding artifacts layer contains structures at larger scales. We use pseudocolor for better visualization.

TABLE II

QUANTITATIVE COMPARISONS ON REAL-WORLD DATASET TWITTER [3]. THE BEST AND THE SECOND BEST RESULTS ARE BOLDFACED AND UNDERLINED

Methods	PSNR	SSIM	PSNR-B
Inputs	27.60	0.7271	27.52
SA-DCT [7]	27.61	0.7281	27.53
LD [34]	27.58	0.7274	27.49
ARCNN [3]	27.54	0.7295	27.49
TNRD [52]	27.60	0.7272	27.52
DnCNN [23]	27.63	0.7294	27.54
LPIO [41]	27.47	0.7333	27.41
MemNet [40]	27.98	0.7441	27.87
KDLS [21]	27.81	0.7371	27.72
DCSC [6]	27.63	0.7313	27.43
IACNN [50]	28.04	0.7424	27.94
RNAN [42]	<u>28.07</u>	<u>0.7459</u>	<u>27.96</u>
Ours	**28.27**	**0.7571**	**28.26**

TABLE III

GENERALIZATION ABILITY ON OTHER TWO JPEG QUALITIES

Methods	QF = 5			QF = 10		
	PSNR	SSIM	PSNR-B	PSNR	SSIM	PSNR-B
Inputs	25.25	0.6570	22.89	32.64	0.9195	29.96
SA-DCT [7]	26.14	0.6816	25.71	32.61	0.9157	30.50
LD [34]	26.07	0.6824	25.48	32.43	0.9172	30.23
ARCNN [3]	26.16	0.6905	25.36	32.83	0.9176	31.05
TNRD [52]	26.23	0.6939	25.58	32.78	0.9158	30.55
DnCNN [23]	**26.41**	0.6972	26.31	34.06	0.9335	32.98
LPIO [41]	25.15	0.6881	24.70	28.55	0.8705	28.41
MemNet [40]	26.33	<u>0.7014</u>	25.55	34.04	0.9330	32.97
KDLS [21]	25.35	0.6986	25.03	33.55	0.9317	32.85
DCSC [6]	26.31	0.7001	**25.73**	34.05	0.9328	**33.03**
IACNN [50]	26.18	0.6883	25.41	33.65	0.9276	32.89
RNAN [42]	26.14	0.6901	<u>25.65</u>	29.38	0.9177	28.98
Ours	<u>26.35</u>	**0.7034**	**25.73**	**34.24**	**0.9344**	<u>33.02</u>

In conventional optimization-based methods, the step size is usually manually defined as a fix number, which limits the model flexibility. Instead, we learn the step sizes $\eta_1^{(t)}$ and $\eta_2^{(t)}$ from the training samples so that they can automatically adjust according to different optimization stages. In Fig. 9, we visualize the learned step sizes at different stages t. Note that the step sizes η_1 for updating $\mathbf{M}$ are relatively small and have flat fluctuations, whereas the step sizes η_2 for updating $\mathbf{O}$

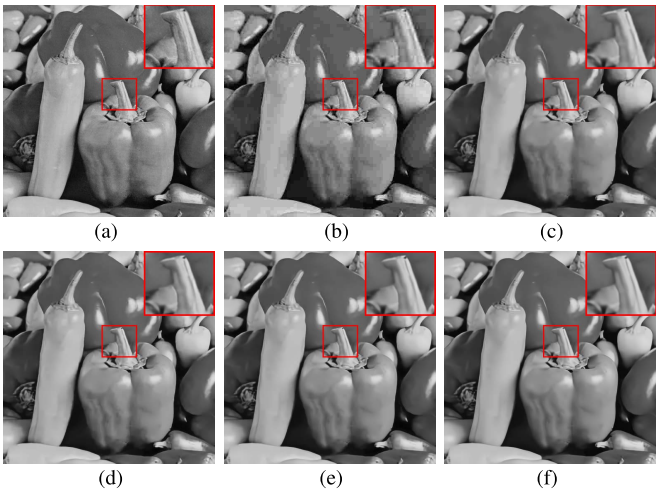


Fig. 8. Visual examples of intermediate $\mathbf{O}^{(t)}$ at different values of t. Please zoomed-in view for better visualization on detail recovery. (a) Clean, PSNR | SSIM | PSNR-B. (b) JPEG, 30.44 | 0.802 | 27.65. (c) $t = 1$, 32.29 | 0.850 | 31.89. (d) $t = 3$, 32.54 | 0.854 | 32.26. (e) $t = 7$, 32.68 | 0.856 | 32.53. (f) $t = 10$, 32.71 | 0.857 | 32.54.

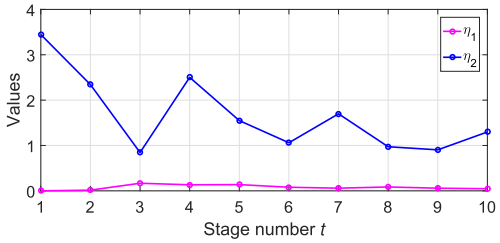


Fig. 9. Learned step sizes $\eta_1^{(t)}$ and $\eta_2^{(t)}$ at different values of t.

have the opposite phenomenon. This is because the sparsity of coefficients $\mathbf{M}$ allows the corresponding optimization process to easily converge to a stable state, so only small step sizes are needed. Obtaining the deblocked image $\mathbf{O}$ requires a process of dense predictions, in which the accumulation of each pixel error will make the total error significantly increase. Therefore, larger step sizes are needed to adjust the descent direction. This visualization helps explain the rationality of the design of learnable step sizes. For training η_1 and η_2, we first define them as variables and empirically initialize them to 0.1 and 1.0, respectively. Then, we utilize backprorogation to automatically update η_1 and η_2 based on the training samples. After training, η_1 and η_2 are fixed and used for testing.

D. Parameters and Running Time

To compare the running time, we test different methods when processing 512×512 grayscale images. Specifically, we test 100 images and record the average running time. As shown in Table IV, although RNAN and our model have relatively high parameter numbers, our running time is less than RNAN and MemNet. This is because RNAN adopts nonlocal operations, which requires the calculation of a large measurement matrix, whereas MemNet deploys multiple recursive modules, which requires a lot of memory. During the testing phase, our network is almost a straightforward feedforward process. This makes our model achieve a good tradeoff between effectiveness and efficiency.

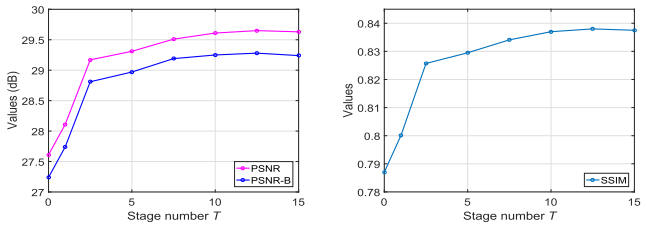


Fig. 10. Average PSNR, PSNR-B, and SSIM values with different stage numbers T.

To isolate the benefit of our proposed model, we further conduct an ablation study by adjusting DnCNN,[1] RNAN,[2] and our model to make their parameter numbers close. We directly use the training and testing code provided by the authors and only adjust the number of convolution kernels while keeping the default network structure unchanged. Specifically, the kernel numbers of DnCNN are set as 80, 140, and 180. The kernel numbers of RNAN are set as 24, 36, and 48. The kernel numbers of our network are set as 16, 24, and 32. All three models are retrained and the quantitative results are shown in Table V. As can be seen, under different orders of magnitude, our method consistently outperforms the other two compared methods.

E. Ablation Studies

We provide ablation studies to explore the effect of each part of our model over the LIVE1 dataset at QF $= 10$.

1) Effect of Stage T: We first test the effect of stage number T and show the PSNR, SSIM, and PSNR-B results in Fig. 10. Using $T = 1$ as a baseline, it is clear that the restoration performance has an obvious improvement with two stages. This demonstrates the effectiveness of our proposed unfolding network to extract rich prior. When $T = 15$, the quantitative results show a slight decreasing trend, which may be caused by the difficulty of gradient propagation due to the increased stage. Therefore, we set $T = 10$ as the default stage number based on this experiment.

2) Effect of the ResNet Depth: In this section, to test the effect of ResNet depth, we set the depth to 4, 6, 8, 10, and 12 separately. The quantitative results are shown in Table VI. In general, a deeper network means a more complex mapping function [18], which can obtain richer features and higher performance. However, keep increasing the depth eventually brings limited improvement, as shown in Table VI. Therefore, we select the ResNet depth as 10 in our experiments.

3) Effect of Regularization Terms: Since the regularization terms play a vital role in solving the ill-posed inverse problem, we conduct two experiments to analyze the influence of the regularization terms.

First, under the current proximal gradient framework, we compare with the scenarios by using three proximal operators as the regularization terms $f_1(\cdot)$ and $f_2(\cdot)$. Specifically, we test the ℓ_1 sparsity prior [26] for conventional prior, the learnable piecewise linear function proposed by ADMM-CSNET [53] for the deep network approach, and

[1] https://github.com/cszn/KAIR/
[2] https://github.com/yulunzhang/RNAN/tree/master/CAR/code

TABLE IV
PARAMETERS AND TIME COMPARISONS ON 100 GRAYSCALE IMAGES WITH A SIZE OF 512×512

Method	ARCNN [3]	DnCNN [23]	LPIO [41]	MemNet [40]	KDLS [21]	DCSC [6]	IACNN [50]	RNAN [42]	Ours
# Params ($\times 10^5$)	1.06	6.69	13.94	6.67	6.53	3.21	43.21	74.09	104.9
Runime (ms)	30.4	50.7	15.9	240.3	47.6	18.3	37.8	1642.7	186.3

TABLE V
QUANTITATIVE COMPARISONS UNDER DIFFERENT PARAMETER NUMBERS

# Params	Methods	PSNR	SSIM	PSNR-B
$\approx 1 \times 10^6$	DnCNN	29.23	0.8129	28.92
	RNAN	29.30	0.8248	28.98
	Our	29.35	0.8274	29.04
$\approx 3 \times 10^6$	DnCNN	29.27	0.8154	28.96
	RNAN	29.37	0.8284	29.06
	Our	29.49	0.8317	29.15
$\approx 5 \times 10^6$	DnCNN	29.36	0.8172	29.03
	RNAN	29.46	0.8307	29.10
	Our	29.54	0.8353	29.19

TABLE VI
EFFECT OF THE RESNET DEPTH IN $\text{proxNet}_{\mathbf{M}}(\cdot)$ AND $\text{proxNet}_{\mathbf{O}}(\cdot)$

Depth	4	6	8	10 (default)	12
PSNR	28.69	29.07	29.49	29.61	29.68
SSIM	0.8142	0.8276	0.8313	0.8370	0.8381
PSNR-B	28.27	28.73	29.16	29.25	29.32

TABLE VII
QUANTITATIVE RESULTS USING DIFFERENT PROXIMAL OPERATORS

Proximal operators	PSNR	SSIM	PNSR-B
ℓ_1-norm [26]	27.48	0.7932	26.12
ADMM-CSNET [53]	27.79	0.8057	26.59
ResNet [18]	29.61	0.8370	29.25
ResNet + ℓ_1-norm	29.52	0.8339	29.27
ResNet + ADMM-CSNET	30.06	0.8402	29.63

our default ResNet. The first three rows of Table VII show the quantitative comparisons and using our default ResNet achieves the best results. This is because compared with the other two single-layer nonlinear activations, the use of ResNet to perform proximal operators not only provides multilayer nonlinear transformations but also introduces extra convolutional operations. This further increases the representation ability of the deep model. For a fair comparison, we use these two functions to replace ReLU, and the last two row of Table VII shows the quantitative comparisons. It is clear that using the learnable piecewise linear function brings significant improvement. This is because the piecewise linear function can approximate any continuous function, and a more flexible nonlinear activation function can thus be learned from training data.

Then, we conduct an ablation study on the architecture of proxNet(·). Specifically, we compare our default ResNet with other two popular network structure, i.e., densely connected network (DenseNet) [19] and deep layer aggregation (DLA) [71]. For a fair comparison, we adjust the three networks to make their parameter numbers close. Table VIII shows the quantitative comparisons, and using DLA architecture achieves the best deblocking performance. This is because DLA adopts an iterative and hierarchical aggregation strategy to merge features. Compared with other two network architectures, DLA is able to better fuse information across layers, which leads to a better performance. Since this work mainly focuses on how to effectively connect model-driven methods and deep learning, we simply choose the ResNet architecture to construct proxNet(·). We believe that other advanced network architectures can also be utilized to further improve the performance.

TABLE VIII
QUANTITATIVE RESULTS USING DIFFERENT NETWORK ARCHITECTURES

Proximal operators	PSNR	SSIM	PNSR-B
ResNet [18]	29.61	0.8370	29.25
DenseNet [19]	29.84	0.8424	29.32
DLA [71]	30.12	0.8462	29.68

TABLE IX
EFFECT OF THE DILATED FACTORS (DF) ON THE PERFORMANCE OF THE PROPOSED MODEL

$DF =$	{1}	{1, 2}	{1, 2, 4} (default)	{1, 2, 4, 6}
PSNR	29.13	29.47	29.61	29.67
SSIM	0.8288	0.8341	0.8370	0.8375
PSNR-B	28.84	29.16	29.25	29.28

4) Effect of Dilated Factors: We also test the impact of dilated factors DF. Specifically, we set the dilated factors DF $\in \{1, 2, 4, 6\}$ while fixing other parameters. Quantitative results are shown in Table IX, and it is clear that increasing dilated factors can generate higher performance. Adding dilated factor results in larger receptive fields, which helps to capture multiscale spatial representations at different compression levels. However, increasing DF will consume more computing resources due to the larger range of convolution operations. Therefore, to balance the tradeoff between performance and speed, we choose DF $\in \{1, 2, 4\}$ as our default setting.

5) Effect of Loss Functions: We also study the effect of loss function with different tradeoff parameters α and β. The quantitative results are shown in Table X. As can be seen, using only a single loss on $\mathbf{O}$, i.e., $\alpha^{(T)} = 1$, can already generate better results than most compared methods, as shown in Table I. Adding $\beta^{(T)} = 1$ can further boost the performance by giving supervision on $\mathbf{H}$, which indicates the importance of imposing supervision on the artifact layer. In addition, using $\alpha^{(t)}$ and $\beta^{(t)}$ to provide supervision on intermediate results helps to optimize the network in a better direction and thus leads to better results. Moreover, we observe that supervision for the artifact layer plays more important role in deblocking performance than that for the image layer, which also reflects the effectiveness of estimating artifacts $\mathbf{H}$. Based on the above analysis, we choose the parameter setting of the last row in Table X as our default loss function.

To further analyze the impact of different loss functions on the deblocking quality, we adopt two different

TABLE X

QUANTITATIVE COMPARISONS ON DIFFERENT LOSS COMBINATIONS ($t \neq T$)

Setting	PSNR	SSIM	PNSR-B
$\alpha^{(T)}=1,\beta^{(T)}=0,\alpha^{(t)}=0,\beta^{(t)}=0$	29.40	0.8317	29.02
$\alpha^{(T)}=1,\beta^{(T)}=1,\alpha^{(t)}=0,\beta^{(t)}=0$	29.50	0.8361	29.17
$\alpha^{(T)}=1,\beta^{(T)}=1,\alpha^{(t)}=0.1,\beta^{(t)}=0$	29.54	0.8360	29.19
$\alpha^{(T)}=1,\beta^{(T)}=1,\alpha^{(t)}=0,\beta^{(t)}=0.1$	29.57	0.8365	29.21
$\alpha^{(T)}=1,\beta^{(T)}=1,\alpha^{(t)}=0.1,\beta^{(t)}=0.1$	29.61	0.8370	29.25

TABLE XI

QUANTITATIVE RESULTS USING MAE AND SSIM LOSSES

Loss setting	PSNR	SSIM	PNSR-B
MAE	29.61	0.8370	29.25
SSIM	29.24	0.8459	28.93
MAE + SSIM	29.43	0.8394	29.06

TABLE XII

QUANTITATIVE RESULTS ADDING THE BCE LOSS

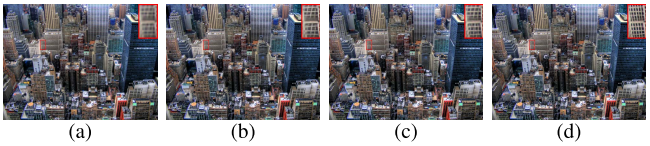
Loss setting	PSNR	SSIM	PNSR-B
MAE	29.61	0.8370	29.25
MAE + BCE	29.87	0.8396	29.49

(a) (b) (c) (d)

Fig. 11. Visual results of adding the BCE loss. (a) JPEG. (b) Clean. (c) MAE. (d) MAE + BCE.

image quality-related penalties, i.e., pixel-level SSIM loss and semantic-level binary cross-entropy (BCE) loss, and add them to the training process. We first test the effect of using different pixel-level loss functions, and the quantitative results are shown in Table XI. It is clear that compared with MAE loss, using SSIM loss increases the SSIM values at the expense of reducing PSNR values. This is because the SSIM loss tends to preserve structural information [72]. By combining MAE and SSIM losses, the deblocked achieves a good tradeoff between PSNR and SSIM.

We also test the BCE loss to see whether semantic information from high-level vision tasks can be used to help image deblocking. Specifically, we first train a VGG-16 classifier [73] by using BCE to distinguish JPEG image (labeled 1) from its clean version (labeled 0). When the output of the trained classifier is equal or close to 0, it indicates that the input is a clean image. In this way, minimizing the prediction of the trained classifier can be regarded as a semantic constraint on the training process. Then, the trained classifier is frozen and used as a new loss and combined with the MAE loss. In Table XII, we show the quantitative results by adding the BCE loss to the training process. It is clear that adding the BCE loss can further improve the deblocking quality. We also show one visual result in Fig. 11 to demonstrate the effect of using the BCE loss. We can find that the deblocked result using the combined loss preserves more details with better visual quality. Other advanced semantic-level loss functions, such as GANs [74] loss and perceptual loss [75], can also be utilized to further improve the deblocking performance.



(a) (b) (c) (d)

Fig. 12. SR results with a scaling factor = 3. (a) Input. (b) DnCNN. (c) RNAN. (d) Our.

F. Extension

Our model is perhaps more general than we presented it as being. Here, we extend our model to one typical image restoration task, i.e., SR. For the SR task, (4) can be modified as

$$\mathbf{J} = \text{Down}(\mathbf{O}) + \mathbf{H} = \text{Down}(\mathbf{O}) + \sum_{n=1}^{N} W_n \otimes M_n \quad (25)$$

where Down$(\cdot)$ denotes the spatial downsampling operator. Then, the objective for SR can be modeled as

$$\arg\min_{\mathbf{O},\mathbf{M}} \left\| \mathbf{J} - \text{Down}(\mathbf{O}) - \sum_{n=1}^{N} W_n \otimes M_n \right\|_F^2 + \lambda_1 f_1(\mathbf{M}) + \lambda_2 f_2(\mathbf{O}). \quad (26)$$

It is clear that the above objective can also be solved by using our proposed unfolding network. Fig. 12 shows one visual comparison on a scale $\times 3$. As can be seen, compared with DnCNN and RNAN, which are also designed for general image restoration tasks, our modified model is able to recover clearer structures. This extension demonstrates that our proposed approach has potential value for other applications.

V. CONCLUSION

In this work, we propose a novel deep unfolding network for JPEG artifacts removal. Different from most deep learning-based methods, which directly build black-box network architectures, we investigate a convolutional model for JPEG artifacts removal and develop an optimization algorithm to solve the model. This optimization algorithm is further unfolded into a deep network, in which the desired images and compression artifacts are alternately estimated. Since each network module is constructed by explicitly following operators of the algorithm, our network has a clear physical meaning with good interpretability. This helps to understand how the entire network works. Moreover, the dilated convolution is embedded into our deep network to explore multiscale redundancies of JPEG compression artifacts. Extensive experiments on both synthetic and real-world datasets validate the superiority of our model over several state-of-the-art methods.

REFERENCES

[1] G. K. Wallace, "The JPEG still picture compression standard," *Commun. ACM*, vol. 34, no. 4, pp. 30–44, Apr. 1991.

[2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[3] C. Dong, Y. Deng, C. C. Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 576–584.

[4] J. Guo and H. Chao, "One-to-many network for visually pleasing compression artifacts reduction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3038–3047.

[5] B. Zheng, Y. Chen, X. Tian, F. Zhou, and X. Liu, "Implicit dual-domain convolutional network for robust color image compression artifact reduction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 11, pp. 3982–3994, Nov. 2020, doi: 10.1109/TCSVT.2019.2931045.

[6] X. Fu, Z.-J. Zha, F. Wu, X. Ding, and J. Paisley, "JPEG artifacts reduction via deep convolutional sparse coding," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2501–2510.

[7] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images," *IEEE Trans. Image Process.*, vol. 16, no. 5, pp. 1395–1411, May 2007.

[8] S. B. Yoo, K. Choi, and J. B. Ra, "Post-processing for blocking artifact reduction based on inter-block correlation," *IEEE Trans. Multimedia*, vol. 16, no. 6, pp. 1536–1548, Oct. 2014.

[9] K. Bredies and M. Holler, "A total variation–based JPEG decompression model," *SIAM J. Imag. Sci.*, vol. 5, no. 1, pp. 366–393, Jan. 2012.

[10] H. Chang, M. K. Ng, and T. Zeng, "Reducing artifacts in JPEG decompression via a learned dictionary," *IEEE Trans. Signal Process.*, vol. 62, no. 3, pp. 718–728, Feb. 2014.

[11] C. Zhao, J. Zhang, S. Ma, X. Fan, Y. Zhang, and W. Gao, "Reducing image compression artifacts by structural sparse representation and quantization constraint prior," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 10, pp. 2057–2071, Oct. 2017.

[12] J. Zhang, R. Xiong, C. Zhao, Y. Zhang, S. Ma, and W. Gao, "CONCOLOR: Constrained non-convex low-rank model for image deblocking," *IEEE Trans. Image Process.*, vol. 25, no. 3, pp. 1246–1259, Mar. 2016.

[13] X. Zhang, W. Lin, R. Xiong, X. Liu, S. Ma, and W. Gao, "Low-rank decomposition-based restoration of compressed images via adaptive noise estimation," *IEEE Trans. Image Process.*, vol. 25, no. 9, pp. 4158–4171, Sep. 2016.

[14] X. Zhang, R. Xiong, X. Fan, S. Ma, and W. Gao, "Compression artifact reduction by overlapped-block transform coefficient estimation with block similarity," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 4613–4626, Dec. 2013.

[15] X. Liu, G. Cheung, X. Wu, and D. Zhao, "Random walk graph Laplacian-based smoothness prior for soft decoding of JPEG images," *IEEE Trans. Image Process.*, vol. 26, no. 2, pp. 509–524, Feb. 2017.

[16] X. Liu, G. Cheung, X. Ji, D. Zhao, and W. Gao, "Graph-based joint dequantization and contrast enhancement of poorly lit JPEG images," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1205–1219, Mar. 2019.

[17] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NeurIPS*, 2012, pp. 1097–1105.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[19] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[20] X. Fu, B. Liang, Y. Huang, X. Ding, and J. Paisley, "Lightweight pyramid networks for image deraining," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 1794–1807, Jun. 2020, doi: 10.1109/TNNLS.2019.2926481.

[21] R. Liu, Z. Jiang, X. Fan, and Z. Luo, "Knowledge-driven deep unrolling for robust image layer separation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 5, pp. 1653–1666, May 2020, doi: 10.1109/TNNLS.2019.2921597.

[22] H. Wang, Q. Xie, Q. Zhao, and D. Meng, "A model-driven deep neural network for single image rain removal," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3103–3112.

[23] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.

[24] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, "Multi-level wavelet-CNN for image restoration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 773–782.

[25] Z. Wang, D. Liu, S. Chang, Q. Ling, Y. Yang, and T. S. Huang, "D3: Deep dual-domain based fast restoration of JPEG-compressed images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2764–2772.

[26] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 399–406.

[27] H. C. Reeve and J. S. Lim, "Reduction of blocking effects in image coding," *Opt. Eng.*, vol. 23, no. 1, Feb. 1984, Art. no. 230134.

[28] B. Ramamurthi and A. Gersho, "Nonlinear space-variant postprocessing of block coded images," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-34, no. 5, pp. 1258–1268, Oct. 1986.

[29] S. Minami and A. Zakhor, "An optimization approach for removing blocking effects in transform coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 2, pp. 74–82, Apr. 1995.

[30] A. Norkin *et al.*, "HEVC deblocking filter," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1746–1754, Dec. 2012.

[31] C. Wang, J. Zhou, and S. Liu, "Adaptive non-local means filter for image deblocking," *Signal Process., Image Commun.*, vol. 28, no. 5, pp. 522–530, May 2013.

[32] J. Ren, J. Liu, M. Li, W. Bai, and Z. Guo, "Image blocking artifacts reduction via patch clustering and low-rank minimization," in *Proc. Data Compress. Conf.*, Mar. 2013, p. 516.

[33] D. Sun and W.-K. Cham, "Postprocessing of low bit-rate block DCT coded images based on a fields of experts prior," *IEEE Trans. Image Process.*, vol. 16, no. 11, pp. 2743–2751, Nov. 2007.

[34] Y. Li, F. Guo, R. T. Tan, and M. S. Brown, "A contrast enhancement framework with JPEG artifacts suppression," in *Proc. ECCV*, 2014, pp. 174–188.

[35] X. Hu, G. Feng, S. Duan, and L. Liu, "A memristive multilayer cellular neural network with applications to image processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 8, pp. 1889–1901, Aug. 2017.

[36] R. Dian, S. Li, and L. Fang, "Learning a low tensor-train rank representation for hyperspectral image super-resolution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2672–2683, Sep. 2019, doi: 10.1109/TNNLS.2018.2885616.

[37] Y. Tang, W. Gong, X. Chen, and W. Li, "Deep inception-residual Laplacian pyramid networks for accurate single-image super-resolution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 5, pp. 1514–1528, May 2020, doi: 10.1109/TNNLS.2019.2920852.

[38] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Proc. NeurIPS*, 2016, pp. 2810–2818.

[39] L. Cavigelli, P. Hager, and L. Benini, "CAS-CNN: A deep convolutional neural network for image compression artifact suppression," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 752–759.

[40] Y. Tai, J. Yang, X. Liu, and C. Xu, "MemNet: A persistent memory network for image restoration," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4539–4547.

[41] Q. Fan, D. Chen, L. Yuan, G. Hua, N. Yu, and B. Chen, "A general decoupled learning framework for parameterized image operators," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 33–47, Jan. 2021, doi: 10.1109/TPAMI.2019.2925793.

[42] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, "Residual non-local attention networks for image restoration," in *Proc. ICLR*, 2019, pp. 1–13.

[43] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. CVPR*, 2018, pp. 7794–7803.

[44] L. Galteri, L. Seidenari, M. Bertini, and A. D. Bimbo, "Deep generative adversarial compression artifact removal," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4826–4835.

[45] L. Galteri, L. Seidenari, M. Bertini, and A. D. Bimbo, "Deep universal generative adversarial compression artifact removal," *IEEE Trans. Multimedia*, vol. 21, no. 8, pp. 2131–2145, Aug. 2019.

[46] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Jan. 21, 2020, doi: 10.1109/TPAMI.2020.2968521.

[47] J. Guo and H. Chao, "Building dual-domain representations for compression artifacts reduction," in *Proc. ECCV*, 2016, pp. 628–644.

[48] X. Zhang, W. Yang, Y. Hu, and J. Liu, "DMCNN: Dual-domain multi-scale convolutional neural network for compression artifacts removal," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 390–394.

[49] H. Chen, X. He, L. Qing, S. Xiong, and T. Q. Nguyen, "DPW-SDNet: Dual pixel-wavelet domain deep CNNs for soft decoding of JPEG-compressed images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 711–720.

[50] Y. Kim *et al.*, "A pseudo-blind convolutional neural network for the reduction of compression artifacts," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 4, pp. 1121–1135, Apr. 2020.

[51] N. Kwak, J. Yoo, and S.-H. Lee, "Image restoration by estimating frequency distribution of local patches," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6684–6692.

[52] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1256–1272, Jun. 2017.

[53] Y. Yang, J. Sun, H. Li, and Z. Xu, "ADMM-CSNet: A deep learning approach for image compressive sensing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 3, pp. 521–538, Mar. 2020.
[54] Y. Li, M. Tofighi, J. Geng, V. Monga, and Y. C. Eldar, "Efficient and interpretable deep blind image deblurring via algorithm unrolling," *IEEE Trans. Comput. Imag.*, vol. 6, pp. 666–681, 2020.
[55] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3929–3938.
[56] D. Yang and J. Sun, "Proximal dehaze-net: A prior learning-based deep network for single image dehazing," in *Proc. ECCV*, 2018, pp. 702–717.
[57] S. Gu, D. Meng, W. Zuo, and L. Zhang, "Joint convolutional analysis and synthesis sparse representation for single image layer separation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1708–1716.
[58] H. Zhang and V. M. Patel, "Convolutional sparse and low-rank coding-based image decomposition," *IEEE Trans. Image Process.*, vol. 27, no. 5, pp. 2121–2133, May 2018.
[59] M. Li *et al.*, "Video rain streak removal by multiscale convolutional sparse coding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6644–6653.
[60] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, Jan. 2009.
[61] Y. Yang, J. Sun, H. Li, and Z. Xu, "Deep ADMM-net for compressive sensing MRI," in *Proc. NeurIPS*, 2016, pp. 10–18.
[62] D. Simon and M. Elad, "Rethinking the CSC model for natural images," in *Proc. NeurIPS*, 2019, pp. 1–11.
[63] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proc. ICLR*, 2016, pp. 1–10.
[64] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.
[65] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. Symp. Operating Syst. Design Implement. (USENIX)*, 2016, pp. 265–283.
[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2014, pp. 1–11.
[67] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proc. Int. Conf. Curves Surf.*, 2010, pp. 711–730.
[68] H. R. Sheikh. 2005. *Live Image Quality Assessment Database Release 2*. [Online]. Available: http://live.ece.utexas.edu/research/quality
[69] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[70] C. Yim and A. C. Bovik, "Quality assessment of deblocked images," *IEEE Trans. Image Process.*, vol. 20, no. 1, pp. 88–98, Jan. 2011.
[71] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2403–2412.
[72] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Trans. Comput. Imag.*, vol. 3, no. 1, pp. 47–57, Mar. 2017.
[73] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556
[74] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. NeurIPS*, 2014, pp. 2672–2680.
[75] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. EECV*, 2016, pp. 694–711.

Xueyang Fu received the Ph.D. degree in signal and information processing from Xiamen University, Xiamen, China, in 2018.

He was a Visiting Scholar with Columbia University, New York, NY, USA, sponsored by the China Scholarship Council, from 2016 to 2017. He is currently an Associate Researcher with the Department of Automation, University of Science and Technology of China, Hefei, China. His research interests include machine learning and image processing.

Menglu Wang received the B.E. degree in electronic engineering from Xidian University, Xi'an, China, in 2018. She is currently pursuing the Ph.D. degree with the University of Science and Technology of China (USTC), Hefei, China.

Her research interests include computer vision and image processing.

Xiangyong Cao received the B.Sc. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 2012 and 2018, respectively.

From 2016 to 2017, he was a Visiting Scholar with Columbia University, New York, NY, USA. He is currently an Assistant Professor with the School of Mathematics and Statistics, Xi'an Jiaotong University. His current research interests include statistical modeling and image processing.

Xinghao Ding received the B.S. and Ph.D. degrees from the Department of Precision Instruments, Hefei University of Technology, Hefei, China, in 1998 and 2003, respectively.

He was a Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA, from 2009 to 2011. Since 2011, he has been a Professor with the School of Informatics, Xiamen University, Xiamen, China. His main research interests include machine learning, representation learning, and computer vision.

Zheng-Jun Zha (Member, IEEE) received the B.E. and Ph.D. degrees from the University of Science and Technology of China, Hefei, China, in 2004 and 2009, respectively.

He is currently a Full Professor with the School of Information Science and Technology and the Executive Director of the National Engineering Laboratory for Brain-Inspired Intelligence Technology and Application, University of Science and Technology of China. His research interests include multimedia analysis, retrieval and applications, computer vision, and pattern recognition. He has authored or coauthored over 100 articles in these areas with a series of publications on top journals and conferences.

Dr. Zha was a recipient of multiple paper awards from prestigious multimedia conferences, including the Best Paper Award and Best Student Paper Award at ACM Multimedia. He serves as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.