An End-to-End Compression Framework Based on Convolutional Neural Networks

Feng Jiang, Wen Tao, Shaohui Liu, Jie Ren, Xun Guo, and Debin Zhao, *Member, IEEE*

***Abstract*— Deep learning, e.g., convolutional neural networks (CNNs), has achieved great success in image processing and computer vision especially in high-level vision applications, such as recognition and understanding. However, it is rarely used to solve low-level vision problems such as image compression studied in this paper. Here, we move forward a step and propose a novel compression framework based on CNNs. To achieve high-quality image compression at low bit rates, two CNNs are seamlessly integrated into an end-to-end compression framework. The first CNN, named compact convolutional neural network (ComCNN), learns an optimal compact representation from an input image, which preserves the structural information and is then encoded using an image codec (e.g., JPEG, JPEG2000, or BPG). The second CNN, named reconstruction convolutional neural network (RecCNN), is used to reconstruct the decoded image with high quality in the decoding end. To make two CNNs effectively collaborate, we develop a unified end-to-end learning algorithm to simultaneously learn ComCNN and RecCNN, which facilitates the accurate reconstruction of the decoded image using RecCNN. Such a design also makes the proposed compression framework compatible with existing image coding standards. Experimental results validate that the proposed compression framework greatly outperforms several compression frameworks that use existing image coding standards with the state-of-the-art deblocking or denoising post-processing methods.**

***Index Terms*— Deep learning, compression framework, compact representation, convolutional neural networks (CNNs).**

I. Introduction

IN RECENT years, image compression attracts increasing interest in image processing and computer vision due to its potential applications in many vision systems. The aim of image compression is to reduce irrelevance and redundancy of an image in order to store or transmit the image at low bit rates [1]. Traditional image coding standards [2] (such as JPEG and JPEG2000) attempt to distribute the available bits for every nonzero quantized transform coefficient in the whole image. While the compression ratio increases, the bits per pixel (BPP) decreases as a result of the use of bigger quantization steps, which will cause the decoded image to have blocking artifacts or noises. To overcome this problem, a lot of efforts have been devoted to improving the quality of the decoded image using a post-processing deblocking or denoising method. Zhai *et al.* [3] propose an effective deblocking method for JPEG images through post-filtering in shifted windows of image blocks. Foi *et al.* [4] develop an image deblocking filtering based on shape-adaptive DCT, in conjunction with the anisotropic local polynomial approximation-intersection of confidence intervals technique. Inspired by the success of nonlocal filters and bilateral filters for image processing, several nonlocal filters have been proposed for image deblocking [5]–[7]. Recently, Zhang *et al.* [8] propose a constrained non-convex low-rank model for image deblocking. Although desired performance is achieved, these post-processing methods are very time-consuming because solving the optimal solutions involves computationally expensive iterative processes. Therefore, it is difficult to apply them to practical applications.

Motivated by the excellent performance of convolutional neural networks (CNNs) in low level computer vision [9]–[11] in recent years and the fact that existing image codecs are extensively used across the world, we propose an end-to-end compression framework, which consists of two CNNs and an image codec as shown in Fig. 2. The first CNN, named compact convolutional neural network (ComCNN), learns an optimal compact representation from an input image, which is then encoded using an image codec (e.g., JPEG, JPEG2000 or BPG). The second CNN, named reconstruction convolutional neural network (RecCNN), is used to reconstruct the decoded image with high quality in the decoding end. Existing image coding standards usually consists of transformation, quantization and entropy coding. Unfortunately, the rounding function in quantization is not differentiable, which brings great challenges to train deep neural networks when performing the backpropagation algorithm. To address this problem, we present a simple but effective learning algorithm to train the proposed end-to-end compression framework by simultaneously learning ComCNN and RecCNN to facilitate the accurately reconstruction of the decoded image using RecCNN. An example of image compression is shown in Fig. 1, from which we can see that the proposed framework achieves much better quality with more visual details. In addition, as shown in Fig. 2, the output of ComCNN preserves, and therefore the compact representation can be compressed by an image codec, such as JPEG, JPEG2000 or BPG.

Manuscript received December 29, 2016; revised April 12, 2017 and June 26, 2017; accepted July 20, 2017. Date of publication August 1, 2017; date of current version October 24, 2018. This work was supported in part by the Major State Basic Research Development Program of China (973 Program) under Grant 2015CB351804, in part by the MOE-Microsoft Key Laboratory, Harbin Institute of Technology, and in part by the National Natural Science Foundation of China under Grant 61572155, Grant 61672188, and Grant 61272386. This paper was recommended by Associate Editor L. Lin. *(Corresponding author: Shaohui Liu.)*

F. Jiang, W. Tao, S. Liu, J. Ren, and D. Zhao are with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China (e-mail: shliu@hit.edu.cn).

X. Guo is with Microsoft Research Asia, Beijing 100080, China (email: xunguo@microsoft.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2017.2734838

Fig. 1. Left: the JPEG-coded image (PSNR = 27.33 dB) with QF = 5, where we could see blocking artifacts,ring effects and blurring on the eyes, abrupt intensity changes on the face. Right: the decoded image (PSNR = 31.14 dB) by our proposed compression framework at the same bit rate with the left image, where the compression artifacts vanished and generated more details.

The contributions of this work are summarized as follows:

1) We propose an end-to-end compression framework using two CNNs and an image codec. ComCNN produces a compact representation for encoding using an image codec. RecCNN reconstructs the decoded image, respectively. To our best knowledge, it is the first time to connect the existing image coding standards with CNNs using a compact intermediate representation.
2) We further propose an effective learning algorithm to simultaneously learn two CNNs, which addresses the problem that the gradient can not be passed in the backpropagation algorithm since the rounding function in quantization is not differentiable.
3) The proposed compression framework is compatible with existing image codecs (e.g. JPEG, JPEG2000 or BPG), which makes our method more applicable to other tasks.

The remainder of this paper is organized as follows. Section II presents a brief review of related work. Section III elaborates the proposed compression framework, including the architectures of ComCNN and RecCNN. Section IV illustrates the experimental results, including the training parameters setting and the solutions to train the ComCNN and RecCNN. In Section V, we conclude this paper.

II. Related Work

A. Image Deblocking and Artifacts Reduction

In the literature, there have been some methods proposed to improve the quality of decoded images using post-processing techniques, which can be roughly categorized into deblocking oriented and restoration oriented methods. The deblocking oriented methods focus on removing blocking and ringing artifacts of the decoded images. Yeh *et al.* [12] propose a self-learning based post-processing method for image/video deblocking by formulating deblocking as a morphological component analysis based image decomposition problem. Yoo *et al.* [13] propose a two-step framework for reducing blocking artifacts in different regions based on increment of inter-block correlation, which classifies the coded image into flat regions and edge regions. Liu *et al.* [14] learns sparse representations within the dual DCT-pixel domain, and achieves very promising results. Recently, Dong *et al.* [9] propose a compact and efficient network (ARCNN) for seamless attenuation of different compression artifacts. Innovatively, D3 [11] and DDCN [10] integrate dual-domain sparse coding and the prior knowledge of JPEG, which achieve impressive results.

The restoration oriented methods regard the compression operation as a distortion process and reduce artifacts by modelling the distortion as statistical distribution and approximating the original image. Sun and Cham [15] model the quantization distortion as Gaussian noises and use field of experts as image priors to restore the images. Zhang *et al.* [16], [17] propose to utilize similarity priors of image blocks to reduce compression artifacts by estimating the transform coefficients of overlapped blocks from non-local blocks. Recently, Zhang *et al.* [8] develop a novel algorithm for image deblocking using a constrained non-convex low-rank model, which formulates image deblocking as an optimization problem within maximum a posteriori framework.

In the aforementioned methods, image prior models play important roles in both the deblocking oriented and restoration oriented methods. However, these methods involve computationally expensive iterative processes when solving the optimal solutions with complex formula derivations. Therefore, they may be not suitable for practical applications. In short, all the related methods reviewed above attempt to improve image quality only from the perspective of image post-processing. In other words, the connection between the encoder front-end processing and the decoder back-end processing is ignored. We attempt to jointly optimize the encoder and decoder to improve the compression performance.

B. Image Super-Resolution Based on Deep Learning

Recently, CNNs have been used successfully for image super-resolution (SR) especially when residual learning [18] and gradients-based optimization algorithms [19]–[21] are proposed to train deeper network efficiently. Dong *et al.* propose a CNN based SR method [22] named SRCNN, which consists of three layers: patch extraction, non-linear mapping and reconstruction. Although Dong *et al.* conclude in their paper that deeper networks do not result in better performance in some cases, other researchers argue that increasing depth significantly boosts performance. For example, VDSR [23] shows a significant improvement in accuracy, which uses 20 weight layers. DRCN [24] has a very deep recursive layer (up to 16 recursions) and outperforms previous methods by a large margin.

C. Image Compression Based on Deep Learning

Recently, deep learning has been used both for lossy and lossless image compression and achieves competitive performance. For the lossy image compression, Toderici *et al.* [25] propose a general framework for variable-rate image compression and a novel architecture based on convolutional and deconvolutional LSTM recurrent networks.

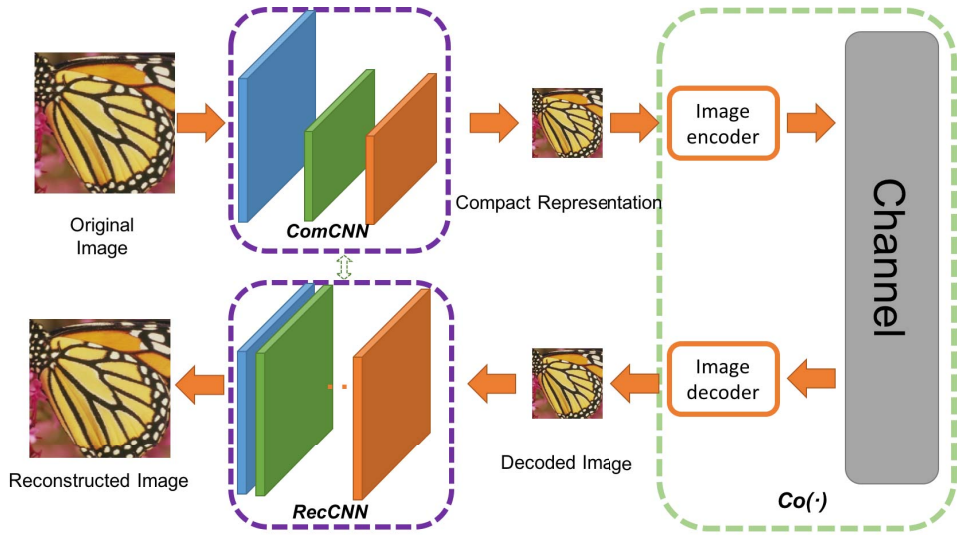


Fig. 2. The proposed novel compression framework. The ComCNN preserves more useful information for reconstruction. Meanwhile, the RecCNN handle the task of reconstructing the decoded image. $Co(\cdot)$ represents an image codec, encoding and decoding the compact representation produced by the ComCNN, which consists of a transform coding stage followed by quantization and entropy coding. In this work, JPEG, JPEG2000 and BPG are adopted.

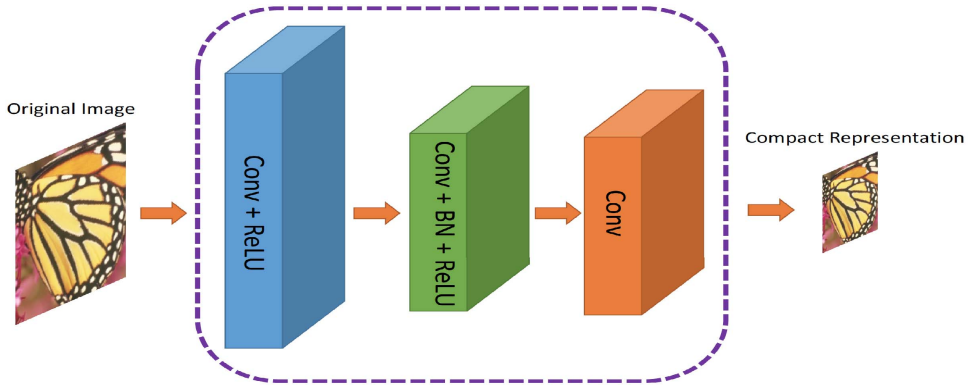


Fig. 3. The architecture of the proposed ComCNN and feature maps of different layers.Upscaled image is obtained by Bicubic interpolation on Decoded image.

Further, Toderici *et al.* [26] propose a neural network which is competitive across compression rates on images of arbitrary size. For a given compression rate, both methods learn the compression models by minimizing the distortion. Theis *et al.* [27] propose compressive autoencoders, which uses a smooth approximation of the discrete of the rounding function and upper-bound the discrete entropy rate loss for continuous relaxation. Ballé *et al.* [28] propose a generalized divisive normalization (GDN) for joint nonlinearity and replace rounding quantization with additive uniform noise for continuous relaxation. Li *et al.* [29] propose a content-weighted compression method with the importance map of image. For the lossless image compression, the methods proposed by Theis and Bethge [30] and A. V. D. Oord *et al.* [31] achieve the best results.

Overall, although the image compression methods based on deep learning achieve competitive performance, they ignored the compatibility with existing image codecs, which limits their use in some existing systems. In contrast, the proposed compression framework takes into account both compression performance and compatibility with existing image codecs.

III. The Proposed Compression Framework

In this section, we first introduce the architecture of the proposed compression framework and then present the detailed learning algorithm.

A. Architecture of End-to-End Compression Framework

As shown in Fig. 2, the proposed compression framework consists of two CNNs and an image codec. The compact representation CNN (ComCNN) is used to generate a compact representation of the input image for the encoding, which preserves structural information of the image and therefore facilitates the accurate reconstruction of high-quality images. The reconstruction CNN (RecCNN) is used to enhance the quality of the decoded image. These two CNNs collaborate with each other and are optimized simultaneously to achieve high-quality image compression at low bit rates.

1) Compact Representation Convolutional Neural Network (ComCNN): As shown in Fig. 3, ComCNN has 3 weight layers, which maintain the spatial structure of the original image and therefore facilitate the accurate reconstruction of

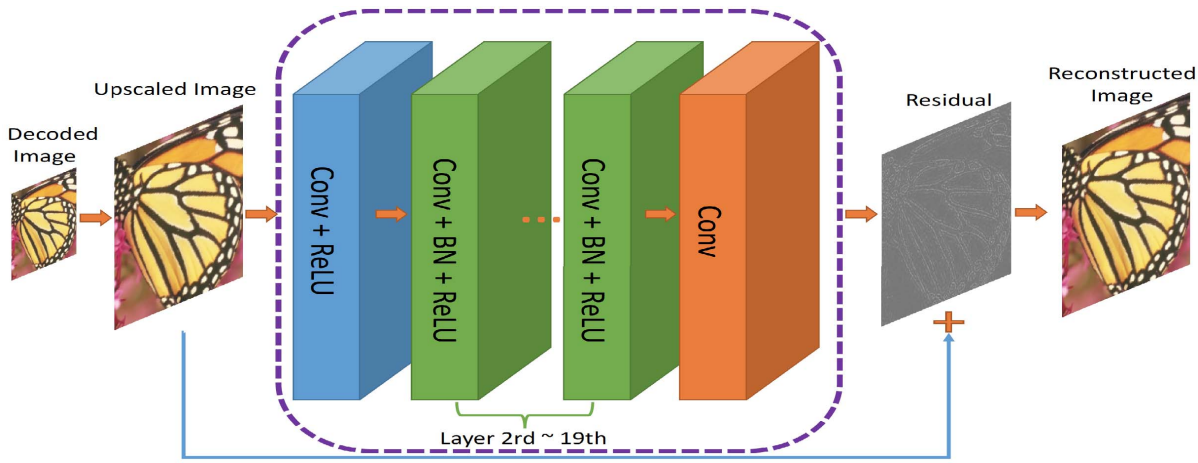


Fig. 4. The architecture of the proposed RecCNN and feature maps of different layers. Upscaled image is obtained by bicubic interpolation on decoded image. The network predicts a residual image, then the sum of the input and the residual gives the high quality output.

Fig. 5. The used test images.

the decoded image using RecCNN.[1] The combination of convolution and ReLU [32] is used in ComCNN. The first layer is used to perform patch extraction and representation which extracts overlapping patches from the input image. Let c represents the number of image channels. A total of 64 filters of size $3 \times 3 \times c$ are used to generate 64 feature maps and the ReLU nonlinearity is utilized as an activation function. The second layer has two significant intentions: downscaling and enhancing the features, which are implemented by convolutional operations with setting the stride to 2. The sizes of 64 filters are $3 \times 3 \times 64$ and ReLU is also applied. For the last layer, c filters of size $3 \times 3 \times 64$ are utilized to construct the compact representation.

2) Reconstruction Convolutional Neural Network (RecCNN): As shown in Fig. 4, RecCNN is composed of 20 weight layers, which have three types of layer combinations: Convolution + ReLU, Convolution + Batch Normalization [33] + ReLU and Convolution. For the first layer, 64 filters of size $3 \times 3 \times c$ are used to generate 64 feature maps, followed by ReLU. For the 2nd to 19th layers, 64 filters of size $3 \times 3 \times 64$ are used, and batch normalization is added between convolution and ReLU. For the last layer, c filters of size $3 \times 3 \times 64$ are used to reconstruct the output. Residual learning and batch normalization are applied to speed up the training process and boost the performance. The compressed image is upsampled to the original image size using bicubic interpolation.

[1]We have tried deeper networks to obtain better performance, but only negligible improvements at the expense of a lot of training time and costs.

B. Learning Algorithm

According to the proposed architecture, both ComCNN and RecCNN try to make the reconstructed image as similar as possible to the original image. Therefore, the end-to-end optimization goal can be formulated as

$$\left(\hat{\theta}_1, \hat{\theta}_2\right) = \arg\min_{\theta_1,\theta_2} \|Re\left(\theta_2, Co\left(Cr\left(\theta_1, x\right)\right)\right) - x\|^2, \quad (1)$$

where x represents the original image, θ_1 and θ_2 are the parameters of ComCNN and RecCNN, respectively. $Cr(\cdot)$ and $Re()$ represent the ComCNN and RecCNN, respectively. $Co(\cdot)$ represents an image codec (e.g. JPEG, JPEG2000 or BPG). From this objective function, we can see that an original image x passes through the compression pipeline, including ComCNN, image codec and RecCNN, and finally outputs the reconstructed image $\hat{x}$. Such a process is an end-to-end compression.

Unfortunately, the $Co(\cdot)$ in Eq.(1) involves a rounding function, which is not differentiable when performing the back propagation algorithm. To solve this problem, we design an iterative optimization learning algorithm. By fixing θ_2, we can get

$$\hat{\theta}_1 = \arg\min_{\theta_1} \left\| Re\left(\hat{\theta}_2, Co\left(Cr\left(\theta_1, x\right)\right)\right) - x \right\|^2, \quad (2)$$

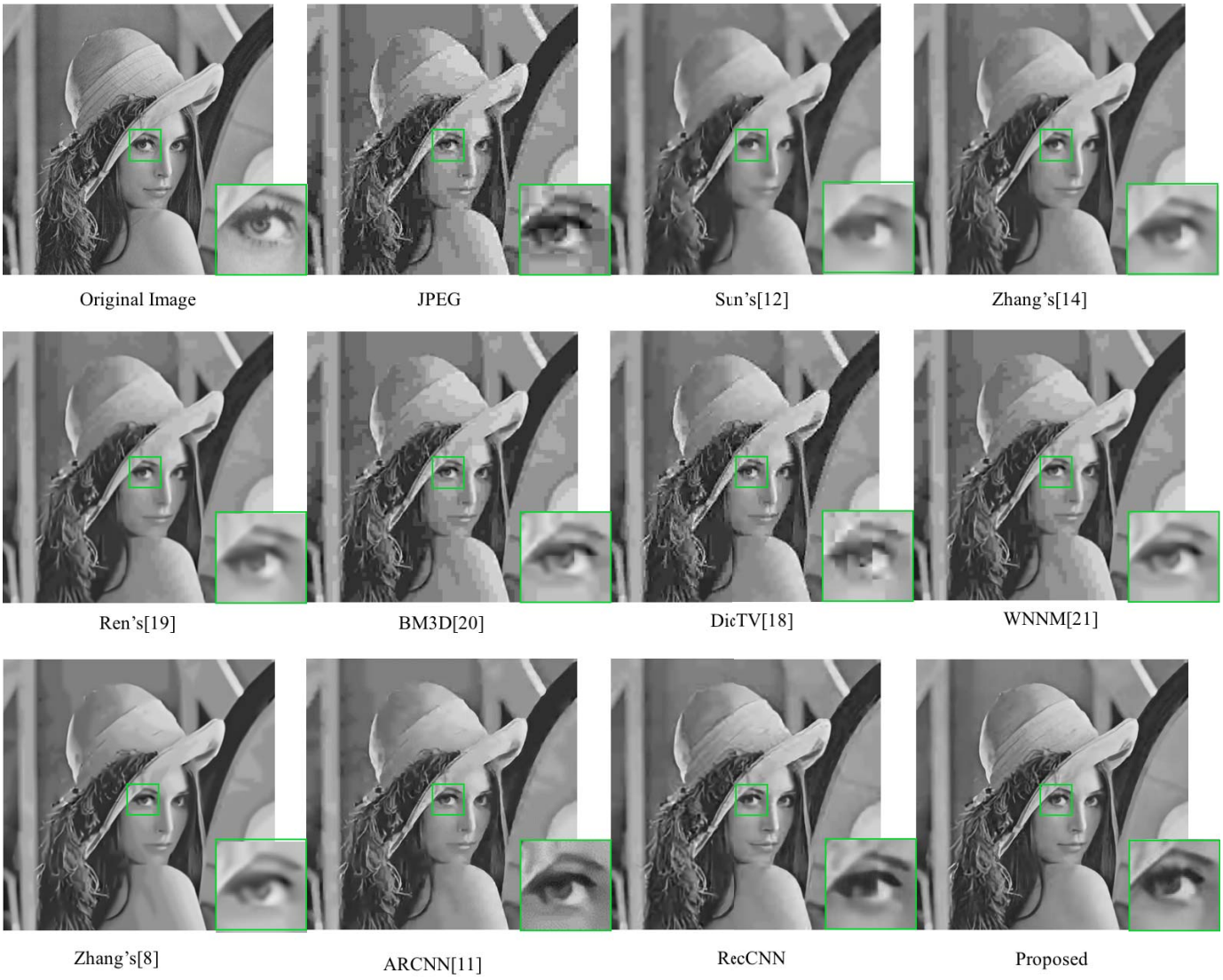


Fig. 6. Visual quality comparison of image deblocking for *Lena* in the case of QF = 5. From left to right and top to bottom: original image, JPEG(PSNR = 27.33 dB, SSIM = 0.7367), the deblocking results by Sun's (PSNR = 28.87 dB, SSIM = 0.8061), Zhang's (PSNR = 29.00 dB, SSIM = 0.8035), Ren's (PSNR = 29.07 dB, SSIM = 0.8010), BM3D (PSNR = 28.63 dB, SSIM = 0.7837), DicTV (PSNR = 28.07 dB, SSIM = 0.7744), WNNM (PSNR = 28.95 dB, SSIM = 0.7947), Zhang's (PSNR = 29.31 dB, SSIM = 0.8169), ARCNN (PSNR = 29.31 dB, SSIM = 0.8142), RecCNN (PSNR = 29.63 dB, SSIM = 0.8195) and the proposed framework (PSNR = 31.14 dB, SSIM = 0.8486).

and by fixing θ_1, we can obtain

$$\hat{\theta}_2 = \arg\min_{\theta_2} \left\| Re\left(\theta_2, Co\left(Cr\left(\hat{\theta}_1, x\right)\right)\right) - x \right\|^2. \quad (3)$$

1) Updating the Parameters θ_2 of RecCNN: According to the network topology, an auxiliary variables $\hat{x}_m$ is introduced and defined as the decoded compact representation of x, which can be formulated as

$$\hat{x}_m = Co\left(Cr\left(\hat{\theta}_1, x\right)\right). \quad (4)$$

After combining Eq.(4) and Eq.(3), we can obtain

$$\hat{\theta}_2 = \arg\min_{\theta_2} \left\| Re\left(\theta_2, \hat{x}_m\right) - x \right\|^2. \quad (5)$$

2) Updating the Parameters θ_1 of ComCNN: From Eq.(2), we can see that it is not a trivial task to obtain the optimal θ_1 since the $Co(\cdot)$ is an inherently non-differentiable operation when performing back propagation. To solve this problem, we define an auxiliary variable $\hat{x}_m^*$ as the optimal input of RecCNN:

$$\hat{x}_m^* = \arg\min_{\hat{x}_m} \left\| Re\left(\hat{\theta}_2, \hat{x}_m\right) - x \right\|^2. \quad (6)$$

Here we make a reasonable and general assumption that $Re\left(\hat{\theta}_2, \cdot\right)$ is monotonic with respect to $\hat{x}_m^*$, which can be expressed as $\left\|\tau - \hat{x}_m^*\right\|^2 \geq \left\|\varphi - \hat{x}_m^*\right\|^2$, if and only if

$$\left\| Re\left(\hat{\theta}_2, \tau\right) - x \right\|^2 \geq \left\| Re\left(\hat{\theta}_2, \varphi\right) - x \right\|^2. \quad (7)$$

TABLE I

JPEG: QF = 5, PSNR (dB) AND SSIM RESULTS OF ALL COMPETITIVE ALGORITHMS GRAYSCALE IMAGE DEBLOCKING AND DENOISING

Test Images	Butterfly	Carmerman	House	Lena	Peppers	Leaves	Parrots	Average
PSNR								
JPEG	22.58	24.45	27.77	27.33	27.17	22.49	26.19	25.43
Sun's [15]	23.83	25.25	29.09	28.87	29.05	23.47	27.45	26.72
Zhang's [17]	24.20	25.39	29.24	29.00	29.07	24.13	27.78	26.97
Ren's [35]	24.58	25.46	29.66	29.07	29.07	24.56	27.87	27.18
BM3D [36]	24.05	25.27	29.21	28.63	28.52	24.02	27.33	26.72
DicTV [34]	23.10	24.54	28.45	28.07	27.95	23.01	26.83	25.99
WNNM [37]	24.75	25.49	29.62	28.95	28.99	24.68	27.80	27.18
Zhang's [8]	25.30	25.61	30.12	29.51	29.61	24.99	28.27	27.63
ARCNN [9]	25.64	25.27	29.68	29.31	29.02	25.07	28.13	27.45
ComCNN	23.05	24.93	29.17	29.46	29.33	23.19	27.67	26.69
RecCNN	25.99	26.33	30.13	29.63	29.81	25.21	28.52	28.02
Proposed	**26.23**	**26.53**	**31.45**	**31.14**	**30.84**	**25.52**	**30.12**	**28.83**
SSIM								
JPEG	0.7378	0.7283	0.7733	0.7367	0.7087	0.7775	0.7581	0.7456
Sun's [15]	0.8321	0.7687	0.8113	0.8061	0.7931	0.8380	0.8323	0.8104
Zhang's [17]	0.8313	0.7672	0.8141	0.8035	0.7895	0.8548	0.8308	0.8130
Ren's [35]	0.8419	0.7666	0.8197	0.8010	0.7876	0.8720	0.8310	0.8171
BM3D [36]	0.8184	0.7607	0.8082	0.7837	0.7639	0.8510	0.8118	0.7997
DicTV [34]	0.7769	0.6658	0.7963	0.7744	0.7456	0.8104	0.8005	0.7671
WNNM [37]	0.8445	0.7674	0.8178	0.7947	0.7827	0.8749	0.8287	0.8158
Zhang's [8]	0.8667	0.7666	0.8285	0.8169	0.8031	0.8882	0.8460	0.8308
ARCNN [9]	0.8741	0.7674	0.8209	0.8142	0.7961	0.8983	0.8446	0.8308
ComCNN	0.7488	0.7662	0.8119	0.8042	0.7966	0.7885	0.8377	0.7934
RecCNN	0.8760	0.7945	0.8251	0.8195	0.8004	0.8803	0.8497	0.8351
Proposed	**0.8847**	**0.8167**	**0.8456**	**0.8486**	**0.8328**	**0.8912**	**0.8951**	**0.8535**

Let $\tilde{\theta}_1$ be the solution of $\arg\min_{\theta_1} \left\| Co\left(Cr\left(\theta_1, x\right)\right) - \hat{x}_m^* \right\|^2$, i.e., for any possible θ_1', it satisfies that

$$\left\| Co\left(Cr\left(\theta_1', x\right)\right) - \hat{x}_m^* \right\|^2 \geq \left\| Co\left(Cr\left(\tilde{\theta}_1, x\right)\right) - \hat{x}_m^* \right\|^2. \quad (8)$$

Following assumption (7), we can obtain that

$$\left\| Re\left(\hat{\theta}_2, Co\left(Cr\left(\theta_1', x\right)\right)\right) - x \right\|^2 \geq \left\| Re\left(\hat{\theta}_2, Co\left(Cr\left(\tilde{\theta}_1, x\right)\right)\right) - x \right\|^2. \quad (9)$$

Accordingly,

$$\tilde{\theta}_1 = \arg\min_{\theta_1} \left\| Re\left(\hat{\theta}_2, Co\left(Cr\left(\theta_1, x\right)\right)\right) - x \right\|^2. \quad (10)$$

Combining with Eq.(2), we can get $\hat{\theta}_1 = \tilde{\theta}_1$, which is

$$\hat{\theta}_1 = \arg\min_{\theta_1} \left\| Co\left(Cr\left(\theta_1, x\right)\right) - \hat{x}_m^* \right\|^2. \quad (11)$$

Since $Co(\cdot)$ is a codec, a reasonable solution of Eq.(11) is

$$\hat{\theta}_1 \approx \arg\min_{\theta_1} \left\| Cr\left(\theta_1, x\right) - \hat{x}_m^* \right\|^2. \quad (12)$$

Combine Eq. (12) and the assumption (7) above, it arrives

$$\hat{\theta}_1 = \arg\min_{\theta_1} \left\| Re\left(\hat{\theta}_2, Cr\left(\theta_1, x\right)\right) - x \right\|^2. \quad (13)$$

Here, we get Eq.(13) with a reasonable assumption and rigorous derivations, which is the approximation of Eq.(2). In this paper, we use Eq.(13) to train ComCNN instead of Eq.(2).

We can obtain the optimal θ_1 and θ_2 by iteratively optimizing Eq.(5) and Eq.(13), respectively. In light of all derivations above, the complete description of the proposed algorithm is given in Algorithm 1.

C. Loss Functions

1) For ComCNN training: Given a set of original images x and trained parameters θ_2, we use mean squared error (MSE) as the loss function

$$L_1(\theta_1) = \frac{1}{2N} \sum_{k=1}^{N} \left\| Re\left(\hat{\theta}_2, Cr\left(\theta_1, x_k\right)\right) - x_k \right\|^2, \quad (14)$$

TABLE II

JPEG: QF = 10, PSNR (dB) AND SSIM RESULTS OF ALL COMPETITIVE ALGORITHMS GRAYSCALE IMAGE DEBLOCKING AND DENOISING

Test Images	Butterfly	Carmerman	House	Lena	Peppers	Leaves	Parrots	Average
PSNR								
JPEG	25.24	26.47	30.56	30.41	30.14	25.40	28.96	28.17
Sun's [15]	26.52	27.26	32.00	31.72	31.62	26.60	30.04	29.39
Zhang's [17]	26.83	27.45	32.11	31.92	31.68	27.26	30.50	29.68
Ren's [35]	27.17	27.43	32.41	31.92	31.63	27.59	30.34	29.78
BM3D [36]	26.64	27.25	32.07	31.77	31.42	26.98	30.05	29.45
DicTV [34]	26.09	26.92	31.77	31.55	31.29	26.33	29.82	29.11
WNNM [37]	27.22	27.40	32.42	31.93	31.64	27.66	30.33	29.80
Zhang's [8]	27.09	**27.72**	33.04	32.19	**31.94**	28.20	30.66	30.12
ARCNN [9]	28.54	27.62	32.53	32.05	31.50	28.31	30.62	30.16
ComCNN	26.32	27.05	31.82	31.63	31.04	26.51	29.97	29.12
RecCNN	28.04	27.33	32.76	32.35	31.34	28.53	30.85	30.17
Proposed	**28.60**	27.44	**33.25**	**33.11**	31.83	**28.77**	**31.11**	**30.59**
SSIM								
JPEG	0.8325	0.7965	0.8183	0.8183	0.7839	0.8609	0.8336	0.8206
Sun's [15]	0.8871	0.8358	0.8504	0.8590	0.8322	0.9138	0.8783	0.8652
Zhang's [17]	0.8923	0.8329	0.8513	0.8597	0.8317	0.9212	0.8804	0.8671
Ren's [35]	0.9010	0.8259	0.8526	0.8571	0.8300	0.9309	0.8775	0.8679
BM3D [36]	0.8896	0.8240	0.8492	0.8549	0.8250	0.9207	0.8749	0.8626
DicTV [34]	0.8699	0.8046	0.8484	0.8559	0.8244	0.9032	0.8741	0.8544
WNNM [37]	0.9019	0.8248	0.8531	0.8571	0.8303	0.9325	0.8775	0.8681
Zhang's [8]	0.9142	0.8401	0.8609	0.8661	0.8358	0.9406	0.8842	0.8774
ARCNN [9]	0.9237	0.8389	0.8591	0.8711	0.8434	**0.9495**	0.8942	0.8828
ComCNN	0.8796	0.8154	0.8497	0.8573	0.8296	0.9095	0.8766	0.8597
RecCNN	0.9192	0.8429	0.8584	0.8679	0.8396	0.9443	0.8884	0.8801
Proposed	**0.9245**	**0.8448**	**0.8678**	**0.8838**	**0.8532**	0.9475	**0.9047**	**0.8895**

Algorithm 1 The Proposed Compression Framework for Training Sub-Networks

Input: The original image x
Initialization: Initializing $\hat{\theta}_1^0$ and $\hat{\theta}_2^0$ as [34]
for $t = 1 \rightarrow T$ **do**
  Update $\hat{x}_m^t$ by computing Eq.(4)
  for $x_m = \hat{x}_m^t$ **do**
    Update $\hat{\theta}_2^t$ by training RecCNN to compute Eq.(5)
  end for
  for $\theta_2 = \hat{\theta}_2^t$ **do**
    Update $\hat{\theta}_1^t$ by training ComCNN to compute Eq.(13)
  end for
end for
Return: $\hat{\theta}_1^t$, $\hat{\theta}_2^t$ and $\hat{x}_m^t$

where N and θ_1 represents the batch size and the trainable parameter, respectively.

2) For RecCNN training: Having obtained a set of compact representation $\hat{x}_m$ from ComCNN and original images x, we use MSE as the loss function:

$$L_2(\theta_2) = \frac{1}{2N}\sum_{k=1}^{N}\left\| res\left(Co\left(\hat{x}_{m_k}\right),\theta_2\right) - \left(Co\left(\hat{x}_{m_k}\right) - x_k\right)\right\|^2, \tag{15}$$

where θ_2 represents the trainable parameter. $res(\cdot)$ represents the residual learned by RecCNN. Clearly, it looks somewhat different from Eq.(5), but they are not contradictory. Actually, they are essentially identical, and Eq.(15) is just expresses Eq.(5) as the form of the residual.

IV. EXPERIMENTS

To evaluate the performance of the proposed compression framework, we conduct experimental comparisons against standard compression methods (e.g., JPEG, JPEG 2000 and BPG) with a post-processing deblocking or denoising method. Five representative image deblocking methods, i.e. Sun'sš [15], DicTV [34], Zhang's [17], Ren'sš [35], Zhang's [8] and two representative image denoising methods, i.e. BM3D [36] and WNNM [37] are chosen due to their state-of-the-art performance. Moreover, ARCNN [9] is also

chosen since it is a landmark deblocking method based on deep learning and achieves the state-of-the-art performance. Meanwhile, in order to demonstrate the effectiveness of ComCNN, we remove ComCNN in the framework and just using RecCNN to reconstruct the decoded image. Similarly, we remove the RecCNN to examine the effect of ComCNN using bicubic interpolation to obtain the reconstructed image of the same size as the original image. The results of all the compared methods are obtained by running the source codes of the original authors with the optimal parameters.

We use the MatConvNet package [38] to train the proposed networks. All experiments are carried out in the Matlab (R2015b) environment running on a computer with Inter(R) Xeon(R) CPU E5-2670 2.60GHz and an Nvidia Tesla K40c GPU. It takes about 28 hours and 24 hours to train the ComCNN and RecCNN on GPU, respectively.

A. Datasets for Training and Testing

Following [39], we use 400 images of size 180×180 for training. A total of 204800 $(400 \times 8 \times 64)^2$ patches are sampled with a stride of 20 on the training images as well as their augmentations (flip and rotate with different angles). Our experiments indicate that using a larger training set can only bring negligible performance improvements. For testing, we use 7 images as shown in Fig. 5, which are widely used in the literature. Please note that all the test images are not included in the training dataset.

B. Model Initialization

We initialize the weights of ComCNN using the method in [40] and use Adam algorithm [21] with $\alpha = 0.001$, $\beta_1 = 0.9, \beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. We train ComCNN for 50 epochs using a batch size of 128. The learning rate is decayed exponentially from 0.01 to 0.0001 for 50 epochs. The weights initialization and gradient updating of RecCNN is the same as ComCNN. RecCNN is also trained for 50 epochs using the same batch size with ComCNN. The learning rate is decayed exponentially from 0.1 to 0.0001 for 50 epochs.

C. Experimental Results

In our experiments, we set different quality factors (QF) to achieve the same bits per pixel (bpp) for both the proposed and compared methods. For the proposed method, we first manually adjust the QF for the compression of compact representation by JPEG to achieve almost the same bpp with the compared image enhancement methods. Then we compare the PSNR and SSIM of the proposed method with the compared methods. The comparison results for all test images with QF = 5 and QF = 10 are provided in Table I and Table II, respectively, where the best results are highlighted in bold.

As seen from Table I and Table II, in the case of QF = 5, the proposed compression framework achieves 1.20dB gains in PSNR and 0.0227 gains in SSIM compared against Zhang's [8], which is state-of-the-art in the compared methods. It is worth mentioning that the proposed framework outperforms all the compared image enhancement methods including ARCNN [9], which is a milestone based on CNN. Meanwhile, in the case of QF = 10, the proposed compression framework achieves 0.43dB and 0.0067 gains in PSNR and SSIM, respectively, compared against ARCNN [9]. The visual quality comparisons in the case of QF = 5 for *Lena* is provided in Fig. 6. We can see that the blocking artifacts are obvious in the image decoded directly by JPEG. DicTV [34], Sun's [15], WNNM [37] and BM3D [36] remove the artifacts partially, but there are still some artifacts visible in the reconstructed image. Zhang's [17] and Ren's [35] generate better results than Sun's [15] and BM3D [36]. However, the blur effects along the edges are generated at the same time. Zhang's [8] achieves a better PSNR and SSIM, but it makes the image over-smoothing and discards some details in image edges. ARCNN [9] and ReCNN achieve better visual quality than other compared methods. The proposed compression framework not only removes most of the artifacts significantly, but also preserves more details on both edges and textures than all the compared methods. In order to verify the effect of ComCNN, we remove ComCNN and use RecCNN alone to reconstruct the decoded image. Similarly, we remove RecCNN and only use ComCNN and bicubic interpolation to examine the effect of RecCNN. As shown in Table I and Table II, worse performances are obtained only with ComCNN or RecCNN. In addition, we show examples of the compact representation produced by ComCNN in Fig. 7. It can be seen that the compact representation maintains the structural information of the original image, but it is different from traditional down sampling methods. In a nutshell, both ComCNN and RecCNN play key roles in the proposed compression framework. Due to the collaboration of ComCNN and RecCNN, the compact representation preserves more useful information

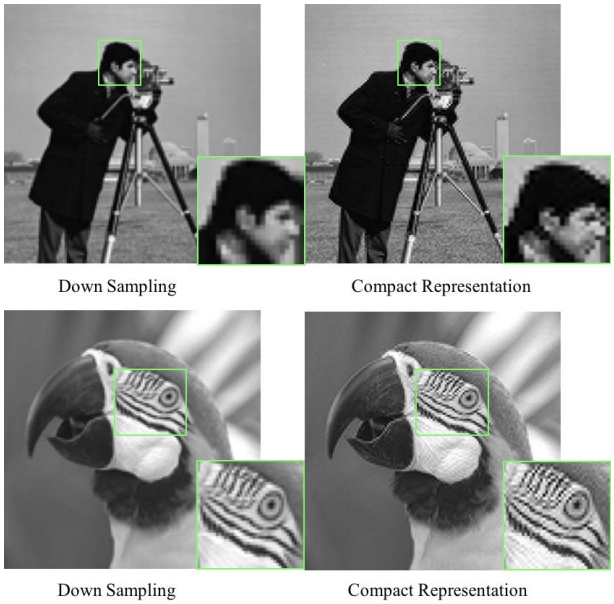


Fig. 7. Visual quality comparison of the down sampling using bicubic interpolation and the compact representation produced by ComCNN.

[2]For each image, there are 8 augmentations and 64 patches of size 40×40 extracted.

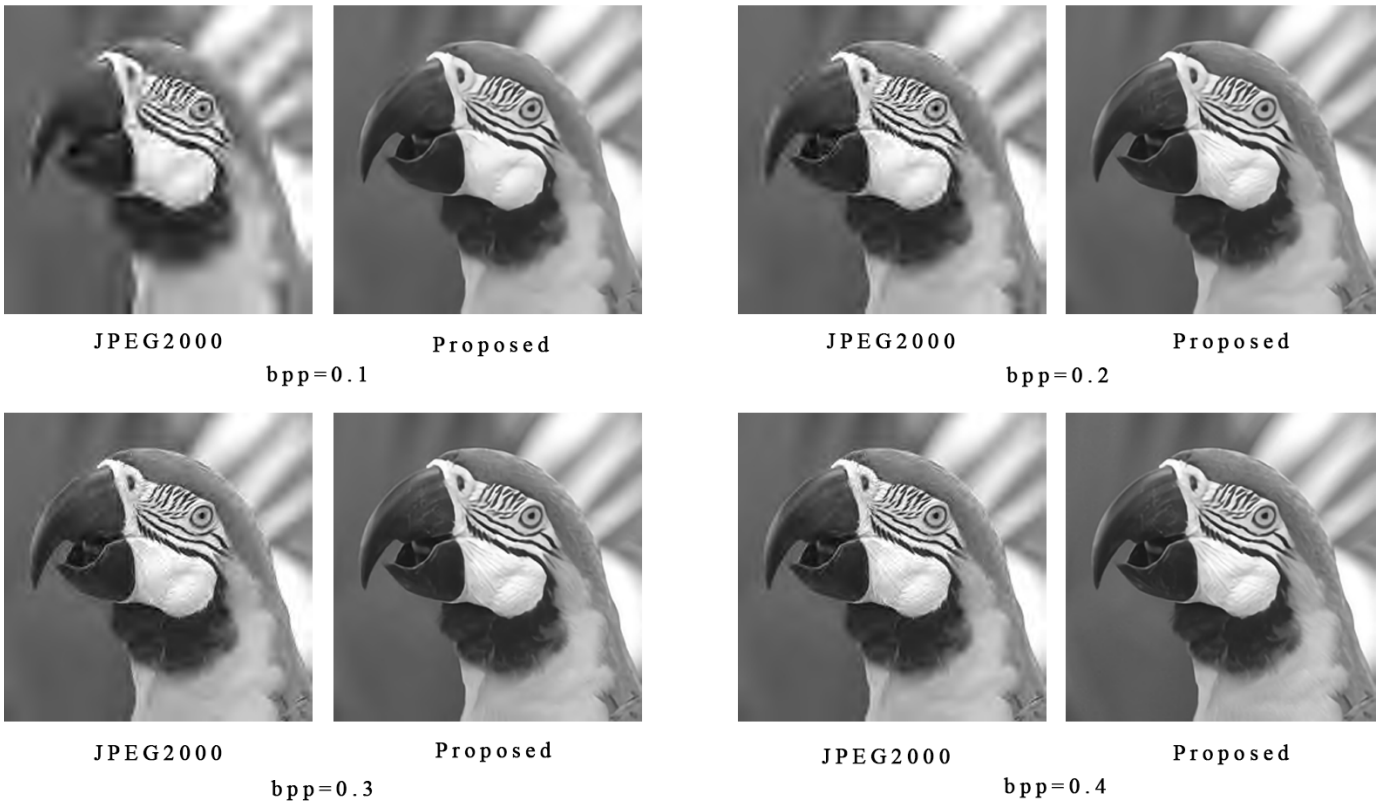


Fig. 8. Subjective performance comparison of JPEG2000 when the bit rate from 0.1bpp to 0.4bpp for *Parrot*. From left to right and top to bottom, the corresponding PSNR(in dB) and SSIM values are (26.05, 0.7853), (31.22, 0.8895), (29.96, 0.8589), (32.72, 0.9242), (32.42, 0.8897), (33.48, 0.9382), (34.42, 0.9150) and (35.09, 0.9480).

TABLE III

JPEG2000: PSNR (dB) AND SSIM RESULTS OF ALL COMPETITIVE ALGORITHMS GRAYSCALE IMAGE DEBLOCKING AND DENOISING

Rate(bpp) / Test Images	0.1		0.2		0.3		0.4	
	JPEG2000	Proposed	JPEG2000	Proposed	JPEG2000	Proposed	JPEG2000	Proposed
PSNR								
Butterfly	18.92	**23.66**	21.99	**26.73**	24.01	**27.47**	25.44	**29.43**
Cameraman	23.33	**26.66**	26.23	**28.44**	28.30	**29.35**	29.80	**30.41**
House	28.02	**30.54**	31.92	**34.34**	34.04	**35.30**	35.35	**35.76**
Lena	29.88	**31.07**	32.98	**34.68**	34.81	**35.65**	36.13	**36.53**
Parrots	26.05	**31.22**	29.96	**32.72**	32.42	**33.48**	34.42	**35.09**
Peppers	29.66	**31.07**	32.56	**33.43**	34.11	**34.49**	35.02	**35.46**
Average	25.98	**27.37**	29.27	**31.72**	31.28	**32.62**	32.69	**33.78**
SSIM								
Butterfly	0.5888	**0.8308**	0.7257	**0.8866**	0.7968	**0.8894**	0.8419	**0.9266**
Cameraman	0.6764	**0.8238**	0.7659	**0.8667**	0.8146	**0.8955**	0.8482	**0.9026**
House	0.7702	**0.8310**	0.8375	**0.8777**	0.8662	**0.8989**	0.8823	**0.9122**
Lena	0.8176	**0.8492**	0.8763	**0.9002**	0.8973	**0.9183**	0.9143	**0.9320**
Parrots	0.7853	**0.8895**	0.8589	**0.9242**	0.8897	**0.9382**	0.9150	**0.9480**
Peppers	0.7851	**0.8273**	0.8368	**0.8733**	0.8591	**0.8988**	0.8719	**0.9127**
Average	0.7273	**0.8419**	0.8169	**0.8878**	0.8540	**0.9065**	0.8789	**0.9224**

for the final image reconstruction. Our testing codes are available in GitHub.[3]

We also evaluate our framework on JPEG 2000 and BPG (Better Portable Graphics)[4], which are shown in Table III and Table IV. BPG compression is based on the Intra Coding of the

[3]https://github.com/compression-framework/compression_framwork_for_tesing

[4]F. Bellard, The BPG Image Format, http://bellard.org/bpg/

TABLE IV
BPG: PSNR (dB) AND SSIM RESULTS OF BPG, BPG + RecCNN AND THE PROPOSED METHOD

Test Images	BPG		BPG + RecCNN		ComCNN + BPG + RecCNN		
	Size(Bytes)	PSNR/SSIM	Size(Bytes)	PSNR/SSIM	Size(Bytes)	PSNR/SSIM	Bits Saving
Cameraman	752	25.95/0.7723	752	26.55/0.7867	642	**26.70/0.7887**	14.63%
	1251	28.12/0.8188	1251	28.77/0.8288	1105	**28.85/0.8325**	11.67%
House	389	29.06/0.8023	389	29.61/0.8121	384	**30.21/0.8274**	1.29%
	592	31.35/0.8349	592	32.02/0.8433	524	**32.31/0.8489**	11.49%
Lena	1616	28.79/0.7896	1616	29.20/0.8016	1585	**29.47/0.8066**	1.92%
	2737	30.97/0.8351	2737	31.46/0.8456	2702	**31.74/0.8529**	1.28%
Butterfly	1277	24.43/0.8335	1277	**25.79/0.8691**	1203	25.65/0.8648	5.79%
	1999	26.86/0.8875	1999	28.03/0.9069	1934	**28.09/0.9076**	3.25%
Peppers	1797	28.93/0.7759	1797	29.59/0.7944	1722	**29.75/0.7978**	4.17%
	2747	30.81/0.8108	2747	31.38/0.8220	2699	**31.43/0.8287**	1.75%
Leaves	1651	24.55/0.8670	1651	25.88/0.9046	1564	**25.97/0.9090**	5.27%
	2492	27.17/0.9195	2492	28.71/0.9405	2423	**28.83/0.9476**	2.77%
Parrots	621	27.94/0.8235	621	28.61/0.8389	595	**28.85/0.8483**	4.19%
	1017	30.25/0.8562	1017	30.95/0.8680	980	**31.19/0.8716**	3.64%
Average	-	28.23/0.8305	-	29.04/0.8473	-	**29.22/0.8523**	5.22%

TABLE V
RUNNING TIME (S) OF COMPARED METHODS IN CPU (/GPU) TESTED ON A 256 × 256 GRAYSCALE IMAGE

Sun's [15]	Zhang's [17]	Ren's [35]	BM3D [36]	DicTV [34]	WNNM [37]	Zhang's [8]	RecCNN	Proposed
132/-	251/-	36/-	3.40/-	53/-	230/-	442/-	1.50/0.015	1.56/0.017

High Efficiency Video Coding (HEVC), which is considered as a major advance in compression techniques. For JPEG2000, we test the proposed compression framework at different bit rates (from 0.1bpp to 0.4bpp) and compare it with JPEG 2000. Table III presents the comparison results with JPEG 2000. It can seen that our framework significantly outperforms JPEG2000 on all test bit-rates of all test images in terms of both PSNR and SSIM. For bpp from 0.1 to 0.4, the proposed framework achieves on average 3.06dB, 2.45dB, 1.34dB, 1.09dB and 0.1047, 0.0709, 0.0525, 0.0435 gains in PSNR and SSIM compared against JPEG2000. In Fig. 8, one can see that the proposed compression framework achieves much better subjective performance than JPEG2000, especially at very low bit rate. Our framework preserves more high-frequency information and recovers sharp edges and pure textures in the reconstructed image. For BPG, we test the BPG codec at QP (quality parameter) = 43 and 47. Further, we keep the bit-rates of the proposed compression framework almost the same for each image. The results are shown in Table IV. One can see that, if we treat RecCNN as a post-processing method, RecCNN achieves on average 0.81dB and 0.0168 gains in PSNR and SSIM. And the proposed compression framework achieves on average 0.99dB and 0.0218 gains in PSNR and SSIM while saving 5.22% bit-rates. It is worth noting that the performance of our proposed compression framework on BPG is not so obvious as on JPEG and JPEG2000, because BPG is already a very good compression method, which might not be significantly improved further.

In order to show the effectiveness of the proposed compression framework, we test our method on Set5 [23], Set14 [23], LIVE1 [41] and General-100 [42] datasets. It is worth mentioning that the General-100 dataset contains 100 bmp-format images with no compression, which are very suitable for compression task. Results are shown in Table VI, from which we can see that the performance of our proposed compression exceeds JPEG and JPEG2000 by a larger margin for all four testing datasets.

D. Running Time

The running time of all compared methods when dealing with a 256 × 256 grayscale image in CPU or GPU are shown in Table V. It should be noted that it is not possible to test the running time in GPU for all other compared methods. As we can see from Table V, the proposed framework needs only 1.56s and 0.017s in CPU and GPU, respectively. Our compression framework is faster than other post-processing methods. In addition, we also calculate the running time of our sub-network RecCNN, which almost takes the entire running time of our method. Because RecCNN has 20 layers, which is much deeper than ComCNN with only 3 layers.

TABLE VI

AVERAGE PSNR(dB)/SSIM RESULTS OF JPEG AND THE PROPOSED METHOD FOR QUALITY FACTORS 5 AND 10, JPEG2000 AND THE PROPOSED METHOD FOR bpp 0.1, 0.2, 0.3 FOR SET5, SET14, LIVE1 AND GENERAL-100

JPEG			
DataSets	Quality Factors	JPEG PSNR/SSIM	Proposed PSNR/SSIM
Set5	5	26.13/0.7206	**29.20/0.8387**
	10	28.99/0.8109	**31.40/0.8854**
Set14	5	24.90/0.6686	**26.89/0.7914**
	10	27.49/0.7762	**28.91/0.8336**
LIVE1	5	24.60/0.6666	**26.78/0.7934**
	10	27.02/0.7720	**28.84/0.8411**
Genreal-100	5	25.93/0.7228	**27.29/0.8447**
	10	28.92/0.8199	**30.16/0.8767**
JPEG2000			
DataSets	Rate(bpp)	JPEG2000 PSNR/SSIM	Proposed PSNR/SSIM
Set5	0.1	26.09/0.7176	**29.00/0.8163**
	0.2	29.11/0.8157	**31.94/0.8859**
	0.3	31.06/0.8629	**33.32/0.9086**
Set14	0.1	25.23/0.6554	**27.88/0.7911**
	0.2	27.80/0.7534	**28.58/0.8273**
	0.3	29.57/0.8063	**30.42/0.8867**
LIVE1	0.1	25.39/0.6612	**27.14/0.7478**
	0.2	27.58/0.7478	**28.54/0.8183**
	0.3	29.19/0.7991	**30.07/0.8544**
General-100	0.1	26.45/0.7179	**27.82/0.8048**
	0.2	29.88/0.8153	**30.86/0.8761**
	0.3	32.00/0.8639	**32.88/0.9083**

V. CONCLUSION

In this paper, we propose an effective end-to-end compression framework based on two CNNs, one of which is used to produce compact intermediate representation for encoding using an image encoder. The other CNN is used to reconstruct the decoded image with high quality. These two CNNs collaborate each other and are trained using a unified optimization method. Experimental results demonstrate that the proposed compression framework achieves state-of-the-art performance and is much faster than most post-processing algorithms. Our work indicates that the performance of the proposed compression framework can be significantly improved by applying the proposed framework, which will inspire other researchers to design better deep neural networks for image compression along this orientation.

REFERENCES

[1] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. 18–34, Feb. 1992.

[2] M. Ghanbari, *Standard Codecs: Image Compression to Advanced Video Coding*. London, U.K.: IET, 2003.

[3] G. Zhai, W. Zhang, X. Yang, W. Lin, and Y. Xu, "Efficient image deblocking based on postfiltering in shifted windows," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 1, pp. 122–126, Jan. 2008.

[4] 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.

[5] R. Zhang, W. Ouyang, and W.-K. Cham, "Image postprocessing by Non-local Kuan's filter," *J. Vis. Commun. Image Represent.*, vol. 22, no. 3, pp. 251–262, Apr. 2011.

[6] N. C. Francisco, N. M. M. Rodrigues, E. A. B. da Silva, and S. M. M. de Faria, "A generic post-deblocking filter for block based image compression algorithms," *Signal Process., Image Commun.*, vol. 27, no. 9, pp. 985–997, 2012.

[7] 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, 2013.

[8] 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.

[9] 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.*, Dec. 2015, pp. 576–584.

[10] J. Guo and H. Chao, "Building dual-domain representations for compression artifacts reduction," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 628–644.

[11] 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.*, Jun. 2016, pp. 2764–2772.

[12] C.-H. Yeh, L.-W. Kang, Y.-W. Chiou, C.-W. Lin, and S.-J. F. Jiang, "Self-learning-based post-processing for image/video deblocking via sparse representation," *J. Vis. Commun. Image Represent.*, vol. 25, no. 5, pp. 891–903, 2014.

[13] 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.

[14] X. Liu, X. Wu, J. Zhou, and D. Zhao, "Data-driven soft decoding of compressed images in dual transform-pixel domain," *IEEE Trans. Image Process.*, vol. 25, no. 4, pp. 1649–1659, Apr. 2016.

[15] 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.

[16] X. Zhang, R. Xiong, S. Ma, and W. Gao, "Reducing blocking artifacts in compressed images via transform-domain non-local coefficients estimation," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2012, pp. 836–841.

[17] 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.

[18] K. He, X. Zhang, S. Ren, and J. Sun. (2015). "Deep residual learning for image recognition." [Online]. Available: https://arxiv.org/abs/1512.03385s

[19] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Feb. 2011.

[20] M. D. Zeiler. (2012). "ADADELTA: An adaptive learning rate method." [Online]. Available: https://arxiv.org/abs/1212.5701

[21] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

[22] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2015.

[23] J. Kim, J. K. Lee, and K. M. Lee. (2015). "Accurate image super-resolution using very deep convolutional networks." [Online]. Available: https://arxiv.org/abs/1511.04587

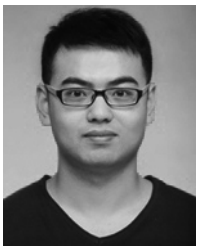
[24] J. Kim, J. K. Lee, and K. M. Lee. (2015). "Deeply-recursive convolutional network for image super-resolution." [Online]. Available: https://arxiv.org/abs/1511.04491

[25] G. Toderici *et al.* (2015). "Variable rate image compression with recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1511.06085

[26] G. Toderici *et al.* (2016). "Full resolution image compression with recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1608.05148

[27] L. Theis, W. Shi, A. Cunningham, and F. Huszár. (2017). "Lossy image compression with compressive autoencoders." [Online]. Available: https://arxiv.org/abs/1703.00395
[28] J. Ballé, V. Laparra, and E. P. Simoncelli. (2016). "End-to-end optimized image compression." [Online]. Available: https://arxiv.org/abs/1611.01704
[29] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang. (2017). "Learning convolutional networks for content-weighted image compression." [Online]. Available: https://arxiv.org/abs/1703.10553
[30] L. Theis and M. Bethge, "Generative image modeling using spatial LSTMs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1927–1935.
[31] A. V. D. Oord, N. Kalchbrenner, and K. Kavukcuoglu. (2016). "Pixel recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1601.06759
[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[33] S. Ioffe and C. Szegedy. (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." [Online]. Available: https://arxiv.org/abs/1502.03167
[34] 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.
[35] 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. (DCC)*, Mar. 2013, p. 516.
[36] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.
[37] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2862–2869.
[38] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. 23rd ACM Int. Conf. Multimedia*, 2015, pp. 689–692.
[39] Y. Chen and T. Pock. (2015). "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration." [Online]. Available: https://arxiv.org/abs/1508.02848
[40] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1026–1034.
[41] H. R. Sheikh, Z. Wang, L. Cormack, and A. C. Bovik. (2005). *LIVE Image Quality Assessment Database Release 2*. [Online]. Available: http://live.ece.utexas.edu/research/quality/
[42] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 391–407.

Feng Jiang received the B.S., M.S., and Ph.D. degrees from the Harbin Institute of Technology (HIT), Harbin, China, in 2001, 2003, and 2008, respectively, all in computer science. He is currently an Associated Professor with the Department of Computer Science, HIT, and a Visiting Scholar with the School of Electrical Engineering, Princeton University. His research interests include computer vision, image and video processing, and pattern recognition.



Wen Tao received the B.S. degrees in computer science from the Harbin Institute of Technology, Harbin, China, in 2016, where he is currently pursuing the M.S. degree with the School of Computer Science and Technology. His current research interests are in image processing and computer vision.

Shaohui Liu received the B.S. degree in computational mathematics and its application software development, the M.Sc. degree in computational mathematics of science, and the Ph.D. degree in computer application technology from the Harbin Institute of Technology, Harbin, China, in 1999, 2001, and 2007, respectively. He was a Visiting Professor and held a post-doctoral position with Sejong University, South Korea, and a Visiting Scholar with the University of Missouri, Columbia, USA. He is currently an Associate Professor with the Harbin Institute of Technology. His research mainly includes image and video processing and analysis, computer vision, multimedia security. In the related fields, he has co-authored over 80 papers, which are cited over 1000 times totally. He is a Senior Member of CCF.

Jie Ren received the B.S. degree from the University of Electronic Science and Technology of China, in 2015. He is currently pursuing the master's degree with the Harbin Institute of Technology. His main research interest includes image processing and multimedia compression technology.

Xun Guo received the Ph.D. degree in computer science from the Harbin Institute of Technology, China, in 2007. From 2007 to 2012, he was with MediaTek Inc., as a Group Manager, where he led a research team and was involved in video compression, especially on technology development for High Efficiency Video Coding standard. He joined Microsoft Research Asia in 2012, where he is currently a Lead Researcher. His research interests include video coding and processing, multimedia system, and computer vision.

Debin Zhao (M'11) received the B.S., M.S., and Ph.D. degrees from the Harbin Institute of Technology (HIT), Harbin, China, in 1985, 1988, and 1998, respectively, all in computer science. He is currently a Professor with the Department of Computer Science, HIT. He has authored over 200 technical articles in refereed journals and conference proceedings in the areas of image and video coding, video processing, video streaming and transmission, and pattern recognition.