DMCNN: DUAL-DOMAIN MULTI-SCALE CONVOLUTIONAL NEURAL NETWORK FOR COMPRESSION ARTIFACTS REMOVAL

Xiaoshuai Zhang, Wenhan Yang, Yueyu Hu, Jiaying Liu †

Institute of Computer Science and Technology, Peking University, Beijing, China

ABSTRACT

JPEG is one of the most commonly used standards among lossy image compression methods. However, JPEG compression inevitably introduces various kinds of artifacts, especially at high compression rates, which could greatly affect the Quality of Experience (QoE). Recently, convolutional neural network (CNN) based methods have shown excellent performance for removing the JPEG artifacts. Lots of efforts have been made to deepen the CNNs and extract deeper features, while relatively few works pay attention to the receptive field of the network. In this paper, we illustrate that the quality of output images can be significantly improved by enlarging the receptive fields in many cases. One step further, we propose a Dual-domain Multi-scale CNN (DMCNN) to take full advantage of redundancies on both the pixel and DCT domains. Experiments show that DMCNN sets a new state-of-the-art for the task of JPEG artifact removal.

Index Terms—Compression Artifacts Removal, Image Restoration, JPEG, Convolutional Neural Network

1. INTRODUCTION

Nowadays, lossy image compression methods (e.g. JPEG, HEVC-MSP and WebP) have been used extensively for image storage and transmission. These methods typically shrink parts of image information by quantization and approximation, so that higher compression rates can be reached. These methods can usually reduce the bit-rate greatly but still maintain satisfactory visual quality by taking advantage of the limitation of the human visual system. But as the compression rate increases, these methods tend to introduce undesirable artifacts such as blocking, ringing, and banding. These artifacts severely degrade the user experience.

In this paper, we examine the degradation of JPEG-compressed images. Typically, a JPEG compressor converts an image of RGB color space into the YCbCr color space. The chroma channels (namely Cb and Cr) are downsampled by the factor of 2. Then, the image is partitioned into $8 \times 8$ blocks and the block-wise 2D Discrete Cosine Transform (DCT) is performed. After DCT, the top left items in each $8 \times 8$ block are low-frequency components, representing the overall features such as the average luminance. The bottom right items are high-frequency components, representing local features such as textures and details. Next, quantization is applied on each of 64 DCT coefficients. As human eyes are not so good at distinguishing high frequency brightness variation, quantization intervals are typically much larger on high-frequency components than low-frequency ones. Noting that the quantization step is the culprit for various kinds of artifacts such as the blocking artifacts within the boundaries of each $8 \times 8$ DCT block, the ringing artifacts around sharp edges, and the noticeable banding effects over the image. As a matter of fact, these kinds of artifacts can be commonly seen on other transform-based methods.

Many methods have been proposed to improve the quality of JPEG-compressed images. Traditional filter-based methods [1, 2] pay attention to general images denoising. Others apply the sparse coding (SC) to restore the compressed images [3, 4]. These methods generally produce sharper images given a compressed input, but they are usually too slow and their results are often accompanied with additional artifacts.

With the rapid development of deep neural networks, multiple CNN-based methods have been used in low-level image processing, including denoising [5, 6], super-resolution [7, 8, 9], video compression [10, 11], rain removal [12, 13, 14]. Specifically for compression artifacts removal, Dong et al. [15] first introduce a CNN-based method and the proposed ARCNN set a good practice for the following low-level CNN-based methods including DnCNN [5], CAS-CNN [16], and MemNet [17]. However, these methods usually work on pixel domain only and do not incorporate much JPEG prior knowledge. More recently, a dual-domain CNN-based model DDCN is proposed in [18]. The model successfully combines DCT-domain prior and the power of the CNN, thus achieves impressive performance.

However, a common weakness of all these CNN-based methods is that the receptive fields of their models are too small, and these models are usually trained on mini image patches (e.g. $35 \times 35$ for ARCNN, $49 \times 49$ for DnCNN, $55 \times 55$ for DDCN), so that only a small range of information (i.e. neighbor pixels) are taken into consideration when
performing the restoration. Noting that local information is sometimes not enough to remove all the artifacts, especially the banding effects, which can appear at a large scale on the image. As shown in Fig. 1, due to the quantization step of JPEG, the sky in the image is split into multiple bandings, and ARCNN fails to recover the bandings because of small receptive fields.

**Fig. 1.** The banding effects can still be clearly seen after the process of ARCNN (QF=10).

To eliminate the banding effects, three efforts have been made to enlarge the receptive fields and enable our model with the ability of extracting global features: (1) Auto-encoder style architecture; (2) Dilated convolutions; (3) Multi-scale loss. Moreover, as validated by DDCN, redundancies in the DCT domain can be effectively utilized. We also adopt a DCT domain branch to enhance the performance of the proposed model.

Our major contribution is to propose an end-to-end CNN with large receptive fields to exploit dual-domain multi-scale features. In order to train the proposed DMCNN more effectively, a modified version of residual learning as well as other training techniques have been utilized. The evaluations on the BSDS500 and the LIVE1 dataset have shown that our work is the current state-of-the-art among all CNN-based JPEG artifact removal networks.

### 2. DUAL-DOMAIN MULTI-SCALE CONVOLUTIONAL NEURAL NETWORK (DMCNN)

The architecture of our proposed DMCNN is in Fig. 2. The model is mainly composed of two similar auto-encoder style networks working on pixel and DCT domains, respectively. The input image is processed by DCT branch first, then passed into the pixel domain branch. The final restoration result is the weighted sum of the input, the DCT branch estimation and the pixel branch estimation.

#### 2.1. Auto-Encoder

The auto-encoder is an efficient way to learn a representation for given data, and typically used for the purpose of dimensionality reduction. An auto-encoder consists of two parts, the encoder $E$ and the decoder $D$, so that:

$$E: \mathcal{X} \rightarrow \mathcal{F},$$
$$D: \mathcal{F} \rightarrow \mathcal{X},$$

$$E, D = \arg \min_{E,D} \|x - (D \circ E)x\|^2, \quad (1)$$

where $\mathcal{X}$ denotes the set of the to-be-compressed data, $\mathcal{F}$ denotes the feature space and $\circ$ is the composition operator. Typically, an auto-encoder is trained with identical image pairs $(x, x) \in \mathcal{X} \times \mathcal{X}$, so that $E$ could learn to extract the best representation of the data in $\mathcal{F}$.

We adopt an auto-encoder style network here as a generative model. Given a compressed image, the encoder $E$ is expected to extract artifact-free features robustly, and then the image is restored from these clean features by decoder $D$.

In order to learn both local and global features, shortcuts are linked between symmetric layers. Also, residual learning strategy is employed. To stress the information learnt form the DCT domain, we add a shortcut with parameter $r$ from the DCT-branch into the final result. Given an input image $C$, the intermediate estimation of the DCT branch $\tilde{O}_D$ and the final output $\tilde{O}_0$ can be formulated as:

$$\tilde{O}_D = \mathcal{D}^{-1}([f(D(C))]_{\text{DRU}}),$$
$$\tilde{O}_0 = g(C, \tilde{O}_D) + r\tilde{O}_D + (1 - r)C, \quad (2)$$

where $\mathcal{D}$ and $\mathcal{D}^{-1}$ stand for the process of $8 \times 8$ block-wise DCT and inverse DCT (IDCT) respectively. $f(\cdot)$ and $g(\cdot)$ denote the processes of the DCT domain auto-encoder and the pixel domain auto-encoder, respectively. $\left[ \cdot \right]_{\text{DRU}}$ denotes the DCT Rectify Unit stated later, and $r$ is a learnable parameter of the residual addition module.

#### 2.2. Dilated Convolution

The dilated convolution is a kind of convolution with predefined gaps, it is first named in [19]. Consider an input image $I$ as a discrete function $I: \mathbb{Z}^2 \rightarrow \mathbb{R}$ and a convolution kernel $k$ shaped $(2r+1) \times (2r+1)$ as a discrete function $k: \Omega_r \rightarrow \mathbb{R}$, where $\Omega_r = [-r, r]^2 \cap \mathbb{Z}^2$. The discrete convolution operator $*_{d}$ with dilation factor $d$ can be defined as:

$$(I *_d k)(p) = \sum_{s + dt = p} I(s)k(t), \quad (3)$$

where $p, s, t \in \mathbb{Z}^2$ are 2D indices.

Unlike normal convolutions, the receptive field of $n$ combined dilated convolutions can reach $(2^n - 1) \times (2^n - 1)$ when dilation factors are set to 1, 2, 4, ..., $2^n - 1$, respectively. The dilated convolution has been widely used in other vision tasks [20, 21] and shown considerable gain, but to the best of our knowledge, has not been used in the task of artifacts removal. In our model, the dilated convolutional layers with dilation factors 2, 4, 8 are used in the middle of the auto-encoder, which aim to enlarge the receptive field further.
With the combination of auto-encoder style architecture and dilated convolutions, the receptive field of our proposed DMCNN reaches $145 \times 145$, which is about 58 times larger than the ARCNN ($19 \times 19$), and 13 times larger than the DnCNN and the DDCN ($41 \times 41$).

2.3. DCT Rectify Unit (DRU)

As stated earlier in the introduction, the main cause of the JPEG compression artifacts is the step of quantization. The quantization step in each of the $8 \times 8$ compression blocks can be formulated as follows:

$$C_{\text{DCT}}(p) = \text{round}(O_{\text{DCT}}(p)/Q(p)) \cdot Q(p), \quad (4)$$

where $C_{\text{DCT}}$ is the quantized DCT block, $O_{\text{DCT}}$ is the original DCT block, $Q$ is the quantization table, and $p \in \mathbb{Z}^2$ is a 2D index. $/ \,$ here denotes the element-wise division.

From (4), it can be easily seen that the estimated $\hat{O}_{\text{DCT}}$ should meet the requirement:

$$C_{\text{DCT}} - Q/2 \leq \hat{O}_{\text{DCT}} \leq C_{\text{DCT}} + Q/2. \quad (5)$$

So like [18] we employ a DCT Rectify Unit (DRU) to constrain the value of DCT block elements, where values out of the range will be cropped. A slight difference is that we drop the leaky slope $\alpha$ in their proposed unit, as no gain can be observed with it. Our DRU can be formulated as:

$$[X]_{\text{DRU}}(p) = \begin{cases} C_{\text{DCT}}(p) - Q(p)/2, & X(p) < C_{\text{DCT}}(p) - Q(p)/2, \\ C_{\text{DCT}}(p) + Q(p)/2, & X(p) > C_{\text{DCT}}(p) + Q(p)/2, \\ X(p), & \text{otherwise.} \end{cases} \quad (6)$$

2.4. Multi-scale DCT-Embedded Loss

As is pointed out by previous works [22] [15], “deeper is not better” in certain low-level tasks. The reason is that deeper neural networks are usually harder to train due to the gradient vanishing. We try to address this problem by redesigning the loss function of the model.

A multi-scale loss is adopted to extract features at different scales. More specifically, features are extracted from different deconvolutional layers of the pixel domain decoder, and scaled images are expected to be reconstructed from these features. By adopting the multi-scale loss, we explicitly guide our network to learn features at different scales.

We also add a DCT loss to train the DCT branch more effectively. Finally, our loss function can be stated as:

$$L(\{\hat{O}_{i=0}\}_{i=0}^2, \{O_{i=0}\}_{i=0}^2) = L_{\text{MMSE}}(\{\hat{O}_{i=0}\}_{i=0}^2, \{O_{i=0}\}_{i=0}^2) + \lambda L_{\text{DCT}}(\hat{O}_D, O_0) \quad (7)$$

where $\{\hat{O}_{i=0}\}_{i=0}^2$ are estimations from pixel domain auto-encoder at different scales, $\{O_{i=0}\}_{i=0}^2$ are original images at different scales, and $\hat{O}_D$ is the intermediate estimation of the DCT branch. Hyper-parameters $\lambda$ and $\theta$ are used to adjust the weights of each loss, they should typically be in the range of $[0, 1]$. $\text{MSE}(\cdot, \cdot)$ denotes the mean squared error loss.

3. EXPERIMENTS

3.1. Implement Details

Datasets. In all experiments, we employ the ImageNet [23] for training. The LIVE1 dataset and the testing set of BSDS500 are used for evaluation. All training and evaluation processes are done on gray-scale images (the Y channel of YCbCr space). The PIL module of python is applied to generate JPEG-compressed images. The module produces numerically identical images as the commonly used MATLAB JPEG encoder after setting the quantization tables manually.

Parameter Settings. The hyper-parameters $\lambda$ and $\theta$ are

![](Fig. 2. The architecture of Dual-domain Multi-scale Convolutional Network (DMCNN).)

Compressed Images
dCT Layer
DCT Domain Auto-Encoder $f$
+ DRU
IDCT Layer
Pixel Domain Auto-Encoder $g$
+$+$
Artifact-free Images
Multi-scale MSE Loss

$X$
CompressedImages
DCTLayer
DCTDomainAuto-Encoder$f$+$+$DRU
IDCTLayer
PixelDomainAuto-Encoder$g$+$+$
Artifact-freeImages
Multi-scaleMSELoss

With the combination of auto-encoder style architecture and dilated convolutions, the receptive field of our proposed DMCNN reaches $145 \times 145$, which is about 58 times larger than the ARCNN ($19 \times 19$), and 13 times larger than the DnCNN and the DDCN ($41 \times 41$).
set to 0.9 and 0.618, respectively. The parameter $r$ of final residual links is initialized to 0.5. The DCT and IDCT layers are fixed and initialized with the corresponding DCT matrix coefficients. Leaky slopes are initialized to 0.1 for PReLU. The depth of the pixel and DCT domain auto-encoders are 15 and 9, respectively.

**Training Details.** Adam optimizer with initial learning rate 0.001 is used for training. The learning rate is scaled down by the factor of 3 when the validation loss stops decreasing. The batch size is set to 80. Training pairs are dynamically generated from the training set. The sizes of training patches are not fixed. As an easy-to-hard transfer, we first train our model on pairs generated with quality factor (QF) of 20 and patch size of $56 \times 56$. Then we gradually increase the patch size till $224 \times 224$. After full convergence, the model dedicated to QF10 is trained based on the previous QF20 model.

### 3.2. Objective Comparisons

To have overall comparisons, we calculate the mean PSNR, SSIM [24], and PSNR-B [25] on the two datasets. We compare to recent state-of-the-arts including pixel domain methods – ARCNN and CAS-CNN, dual domain method – DDCN and general frameworks – TNRD [26] and DnCNN[5].

The quantitative results are shown in Table 1 and Table 2. Generally, our proposed model DMCNN outperforms all the other methods on all evaluated datasets, QFs and metrics. Specifically, our model far surpasses all pixel domain methods and general frameworks. Also, considerable gains can be observed compared to the dual domain method DDCN.

### 3.3. Subjective Comparisons

For subjective comparisons, some restored images of different approaches on the LIVE1 dataset have been presented. As can be seen in Fig. 3, the results of DMCNN are more visually pleasing. Due to the large receptive fields, our model is able to handle quantization banding effects as well as recover lost details using regional patterns. More experiment results are available on our project page ¹.

### 4. CONCLUSIONS

In this paper, we introduce a novel network based on dual-domain auto-encoders, named DMCNN. By applying dilated convolutional layers and multi-scale loss, our model is able to extract global information and eliminate JPEG compression artifacts effectively.

¹http://i.buriedjet.com/projects/DMCNN/
5. REFERENCES


