Learning Dual Priors for JPEG Compression Artifacts Removal

Xueyang Fu¹, Xi Wang¹, Aiping Liu¹, Junwei Han², Zheng-Jun Zha¹*
¹University of Science and Technology of China, China
²Northwestern Polytechnical University, China
{xyfu, aipingl, zhazj}@ustc.edu.cn, wangxxi@mail.ustc.edu.cn, jhan@nwpu.edu.cn

Abstract

Deep learning (DL)-based methods have achieved great success in solving the ill-posed JPEG compression artifacts removal problem. However, as most DL architectures are designed to directly learn pixel-level mapping relationships, they largely ignore semantic-level information and lack sufficient interpretability. To address the above issues, in this work, we propose an interpretable deep network to learn both pixel-level regressive prior and semantic-level discriminative prior. Specifically, we design a variation-al model to formulate the image de-blocking problem and propose two prior terms for the image content and gradient, respectively. The content-relevant prior is formulated as a DL-based image-to-image regressor to perform as a de-blocker from the pixel-level. The gradient-relevant prior serves as a DL-based classifier to distinguish whether the image is compressed from the semantic-level. To effectively solve the variational model, we design an alternating minimization algorithm and unfold it into a deep network architecture. In this way, not only the interpretability of the deep network is increased, but also the dual priors can be well estimated from training samples. By integrating the two priors into a single framework, the image de-blocking problem can be well-constrained, leading to a better performance. Experiments on benchmarks and real-world use cases demonstrate the superiority of our method to the existing state-of-the-art approaches.

1. Introduction

With the rapid development of consumer devices (e.g., digital cameras and smartphones) and wireless network, the number of images and videos has achieved explosive growth, which has brought more pressure and challenges to storage and transmission systems. To save the storage capacity and transmission bandwidth, captured images and videos are usually compressed to reduce information redundancy. Lossy compression algorithms, e.g., Joint Photographic Experts Group (JPEG) [43] and High Efficiency Video Coding (HEVC) [41], have been widely explored to achieve this goal. However, due to the inevitable signal loss during compression, these compression algorithms usually generate visually unpleasing compression artifacts. These artifacts not only decrease the visual quality, but also degrade the performance of downstream computer vision systems, especially at high compression ratios. Therefore, removing compression artifacts is an important post-processing task and has attracted more attention in recent years [20,30]. We refer the reader to review articles [28,30] for more details. In this paper, we focus on alleviating still image degradation caused by JPEG compression, which is one of the most prevalent compression standards.

JPEG compression first applies the discrete cosine transformation (DCT) on 8×8 pixel blocks. Then, these DCT coefficients are coarsely quantized to remove high-frequency details to save space. Due to the independent processing on each pixel block and the removal of high-frequency details, compressed images usually suffer from blocking and blurring artifacts. In addition, using a large quantization step, banding artifacts will appear in smooth areas. Some recent studies have proposed methods to remove undesirable JPEG compression artifacts. According to the design mechanism, these methods can be roughly classified into two categories: model-based methods and deep learning (DL)-based methods. Early model-based works perform filtering to remove compression artifacts. For instance, Foi et al. [12] propose a shape-adaptive DCT filtering method for compression artifacts reduction. On the other hand, since multiple latent clear versions can be estimated from a single compressed image, this task is essentially an ill-posed inverse problem, which requires prior knowledge to constrain it. Along this research direction, many researchers formulate this problem as a minimization of a variational model with favor-
able prior terms. Based on the maximum a posterior (MAP) framework, many prior models, e.g., quantization step [55], sparse representation [3] and low rank [59], have been developed. Although these model-driven methods have shown good performance, the representation abilities of handcrafted priors are limited, which leads to unstable results when processing compressed images with complex structures.

In the past few years, DL-based methods have achieved significant progress of JPEG artifacts removal [8, 57, 63]. Due to the powerful nonlinear capacity [16, 21, 22, 29, 53] and huge amounts of training data, these methods can learn the inverse mapping of compression degradations, and thus produce better results than model-driven methods. However, most of current DL-based methods adopt feed-forward networks to directly predict clear images, making them like black boxes and lack interpretability. In addition, since these DL-based methods only learn pixel-level mappings, semantic-level information is not fully explored and exploited, which further limits their performance improvement.

Different from these methods, we propose an interpretable deep network by combining advantages of both the model-based methods and data-driven DL models. Specifically, we introduce an effective algorithm with DL to learn a pixel-level regressive prior for image content and a semantic-level discriminative prior for image gradient, respectively. First, we model the content-relevant prior as an image-to-image regressor to perform de-blocking, and design the gradient-relevant prior as binary classifier to distinguish whether the image is compressed. Then, the image de-blocking problem is formulated as a minimization of a variational model with the two proposed priors. To effectively solve the model, we design an alternating minimization scheme based on the gradient descent technique and half-quadratic splitting method. Finally, the iterative algorithm is unfolded into a deep network architecture, in which the two priors can be automatically learned through an effective network training strategy. We show that our method is able to predict visually pleasing de-blocked images while removing undesirable JPEG artifacts sufficiently. The contributions of this work are as follows:

- We propose two effective priors to describe the image content and image gradient from the pixel-level and semantic-level, respectively. By using the two priors as the regularizer, we introduce a new variational model for the JPEG compression artifacts removal.

- We propose an alternating minimization algorithm, which is based on the gradient descent technique and half-quadratic splitting method, to solve the variational model. By unfolding the algorithm, we design a new deep network architecture for the image de-blocking problem. In this way, the two proposed priors can be automatically estimated from training samples. In addition, since the feed-forward process mimics the processing flow of the alternating minimization algorithm, the interpretability of the deep model is increased.

- We collect a new dataset containing compressed/clear image pairs based on the popular online social software WeChat. This dataset aims to complement the existing Twitter dataset [8] to serve the relevant research communities. Extensive experiments show that our proposed network performs favorably against state-of-the-arts on both benchmarks and real-world use cases.

2. Related work

2.1. Model-based methods

In early studies, image filtering technologies are widely explored to explicitly remove compression artifacts. For instance, a quadratic programming technology with an adaptive prior is proposed in [36] to remove blocking artifacts and preserve image details. An adaptive neighborhood for smoothing and conducting post-filtering in shifted windows is introduced in method [54]. Yoo et al. [49] utilize group-based filtering to improve the correlation between image blocks and reduce blocking artifacts. Foi et al. [12] achieve both the image de-noising and de-blocking by conducting filtering in the shape-adaptive DCT domain.

On the other hand, prior knowledge also plays a vital role in this task since it is an ill-posed inverse problem. Many researchers make great effort to explore effective priors to constrain the solution space. As one of the most important priors, the quantization step can be utilized to estimate the range of the DCT coefficients of the clear image to constrain the de-blocked result [35, 55]. Other image priors, e.g., sparse representation [3, 4, 40, 52], low rank [59], non-local self-similarity [60] and graph [34], are also explored and exploited. Li et al. [27] combine image decomposition algorithm and sparse prior to achieve both JPEG artifacts removal and image enhancement. Liu et al. [35] improve image de-blocking by exploiting the sparsity in both the image and DCT domains. In [37], a graph-based low-rank prior is introduced to reflect the manifold structures of image patches. Zhang et al. [59] reduce compression artifacts by exploring the non-local similarity in the DCT domain. Liu et al. [34] propose a graph smoothness prior to jointly reduce compression artifacts and improve contrast based on the Retinex theory. Although these model-based methods are flexible and have good interpretability, they usually have limited representation capabilities with handcrafted filters and priors.

2.2. Deep learning-based methods

In the past few years, deep learning has made breakthrough progress in JPEG compression artifacts removal.
Due to the powerful nonlinear representation ability, DL-based methods usually have better performance than model-based ones. Dong et al. [8] introduce the first DL-based method by designing a four-layer CNNs architecture, which makes a breakthrough progress in compression artifacts removal. Inspired by the residual learning, several deep network architectures are well designed for JPEG compression artifacts removal and relevant restoration tasks [18, 32, 42, 57, 62, 63]. Fan et al. [11] construct a decouple learning framework by incorporating different parameterized image operators for various image restoration tasks. To produce visually pleasing results, the generative adversarial networks (GANs) are utilized to capture the underlying data distribution to generate vivid image textures [14, 15, 20]. Yoo et al. [48] achieve image de-blocking by estimating frequency distribution of local patches. Kim et al. [23] propose a pseudo-blind de-blocking method, in which the quality factor is estimated for both blind and non-blind de-blocking. Many dual domain learning-based methods [5, 10, 19, 61] are also introduced by considering the DCT-relevant prior.

Recently, many researchers attempt to combine both the domain knowledge and deep learning for various image restoration tasks. Wang et al. [45] build a dual domain network by using the DCT-pixel domain sparse coding and the learned iterative shrinkage thresholding algorithm. Chen et al. [6] design a deep network based on the classic iterative nonlinear reaction diffusion for effective and efficient image restoration. In [7, 13], the authors adopt the classic convolutional sparse coding to solve the image de-blocking. Yang et al. [46] use sparsely sampled measurements for image reconstruction by combining compressive sensing theory and deep learning. Under the alternative minimization framework, deep CNNs are also utilized to learn priors and perform as the regularizer [9, 31, 33, 56, 58]. Our method shares these similar spirits, but different from the above approaches that only learn pixel-level mapping relationships, we further introduce semantic-level information to better handle the JPEG compression artifacts removal.

3. Methodology

3.1. Motivation

In general, JPEG artifacts removal aims to obtain the clear image $x$ from its compressed observation $y = x + v$, where $v$ contains compression artifacts and residual image content. Since this task is an ill-posed inverse problem, from a Bayesian perspective, the clear image can be obtained by solving a MAP problem:

$$\arg \max_x \log p(y | x) + \log p(x),$$  \hspace{1cm} (1)

where $\log p(y | x)$ and $\log p(x)$ denote the data likelihood and the prior terms, respectively. Formally, by performing a negative logarithmic transformation, Equation (1) can be reformulated as an energy minimization variational model:

$$\arg \max_x ||y - x||_2^2 + \lambda f(x),$$  \hspace{1cm} (2)

where $f(x)$ denotes the regularizer associated with the prior $\log p(x)$, and $\lambda$ is a trade-off parameter. It is clear that the regularizer plays a vital role in obtaining high-quality solutions. In model-based optimization methods, many regularizers have been explored, e.g., low rank [59] and nonlocal self-similarity [60]. While these methods are usually time-consuming with handcrafted priors that are not powerful enough for good performance. Therefore, with the powerful nonlinear capabilities, deep unfolding networks [7, 13] have been explored to extract the priors from training samples.

Albeit improvement in interpretability and performance, existing deep unfolding methods only design pixel-level regression networks for the regularization term, semantic-level information is not fully utilized. Affected by compression artifacts (e.g., blocking, blurring and banding), the overall quality of the JPEG compressed image will be significantly lower than its clear counterpart. In other words, it should be easy to distinguish whether the image has been compressed from the semantic-level. This observation motivates us to introduce a semantic-level discriminative prior to complement existing methods for image de-blocking.

In addition, according to our domain knowledge, most compression artifacts have a greater impact on the high-frequency part than the low-frequency part. As shown in Figure 1, due to the inherent sparsity of gradients, the histogram on the gradient domain has stronger regularity than the image domain. Moreover, the histogram of the compressed gradients $\nabla y$ is much sparser than the clear gradients $\nabla x$, as shown in Figure 1. This is because the quantization intervals are much larger in high-frequency part than those in low-frequency part, which results in significant changes in the high-frequency part before and after compression. Therefore, we argue that using the high-frequency part of the image can provide better discriminative information. Based on the above observation, we design a semantic-level prior and apply it to the image gradient, which is the most commonly used high-frequency image information [38]. By adding the semantic-level discriminative prior to Equation (2), our final variational model is:

$$\arg \max_x ||y - x||_2^2 + \lambda_1 f_p(x) + \lambda_2 f_s(\nabla x),$$  \hspace{1cm} (3)

where $\nabla$ is the differential operator. $f_p(\cdot)$ and $f_s(\cdot)$ mean the regularizers to deliver the pixel-level regressive prior and semantic-level discriminative prior, respectively.

3.2. Optimization

To construct a step-by-step corresponding deep unfolding network architecture for Equation (3), we first design an
effective alternating minimization algorithm to obtain the unfolding inference. By introducing two auxiliary variables $u$ and $g$, Equation (3) can be rewritten as a non-constrained optimization problem:

$$
\arg\max_{x,u,g} \|y - x\|_2^2 + \alpha \|u - x\|_2^2 + \beta \|g - \nabla x\|_2^2 + \lambda_1 f_p(u) + \lambda_2 f_s(g),
$$

where $\alpha$ and $\beta$ are penalty parameters. Equation (4) can be addressed by alternatively solving sub-problems:

$$
\begin{align*}
\arg\max_u \alpha \|u - x_{k-1}\|_2^2 + \lambda_1 f_p(u), \\
\arg\max_g \beta \|g - \nabla x_{k-1}\|_2^2 + \lambda_2 f_s(g), \\
\arg\max_x \alpha \|y - x\|_2^2 + \alpha \|u_k - x\|_2^2 + \beta \|g_k - \nabla x\|_2^2,
\end{align*}
$$

where $k = 1, 2, ..., K$ is the iteration. Below, we detail the updates for each sub-problem.

1) Solving $u$ The sub-problem (5) is a proximity operator of $f_p(\cdot)$ and corresponds to de-block the image $x_{k-1}$. The solution can be expressed as:

$$
u_k = \text{deblocker}(x_{k-1}),$$

where $\text{deblocker}(\cdot)$ can be arbitrary image de-blocking algorithms. In this paper, we design a deep convolutional neural networks to perform the de-blocker. In this way, complicated image content-relevant priors can be directly learned from training data without manual design.

2) Solving $g$ Since we want to introduce semantic-level discriminative information for image de-blocking, $f_s(\cdot)$ is designed as a binary classifier since classification is the most fundamental semantic-related analysis. Therefore, unlike solving $u$ that directly deploys an image-to-image regression network, we follow Li et al. [26] and adopt the back-propagation to compute the derivative of $f_s(\cdot)$. The solution for solving $g$ is:

$$
g_{d,k}^{(j)} = g_{d,k}^{(j-1)} - \eta \left[ \beta (g_{d,k}^{(j-1)} - \nabla_d x_{k-1}) + \lambda_2 \frac{\partial f_s(g_{d,k}^{(j-1)})}{\partial g_{d,k}^{(j-1)}} \right],
$$

where $d \in \{h, v\}$ are the horizontal and vertical directions, respectively. $\eta$ is the step size, and $j$ is the inner iteration. To fit with the overall energy minimization, we intentionally label clear images as 0 (negative) and compressed images as 1 (positive). In this way, during the optimization, Equation (9) provides semantic determining for $g$ whether increasing or decreasing its value will improve the clarity, which complements the pixel-level constraints in Equation (8).

3) Solving $x$ Since Equation (7) is a least squares problem, it has a closed form solution. To speed up the process, we adopt Fast Fourier Transformation (FFT) to diagonalize the differential operator so that large-matrix inversion can be avoided. By setting the first-order derivative to zero, the solution of Equation (7) is:

$$
x_k = F^{-1} \left( \frac{\mathcal{F}(y) + \alpha \mathcal{F}(u_k) + \beta \left( \sum_{d \in \{h,v\}} \mathcal{F}^*(\nabla_d) \mathcal{F}(g_k) \right)}{\mathcal{F}(I) + \alpha \mathcal{F}(I) + \beta \left( \sum_{d \in \{h,v\}} \mathcal{F}^*(\nabla_d) \mathcal{F}(\nabla_d) \right)} \right),
$$

where $I$ is the identity matrix, $\mathcal{F}$ is the FFT operator, $\mathcal{F}^*$ is the complex conjugate operator, $F^{-1}$ is the inverse FFT operator, $\nabla_h$ and $\nabla_v$ are the horizontal and vertical differential operators, respectively. Since all calculations are performed pixel-wise, the update of $x$ can be efficiently computed. However, if complex handcrafted de-blocker and classifier are used, the entire optimization process will be computationally expensive. Therefore, we unfold the above algorithm into a deep network to gain advantages from both model-based optimization and data-driven deep learning.

3.3. Deep unfolding network

As shown in Figure 2, our deep unfolding network contains $K$ stages, which are intentionally designed to correspond to $K$ iterations in the optimization algorithm. In each network stage, two auxiliary variables are first updated, and then the de-blocked image is calculated. Therefore, the question left to us now is how to design the pixel-level regularizer $f_p(\cdot)$ and the semantic-level regularizer $f_s(\cdot)$.

De-blocker $f_p(\cdot)$ To achieve an image-to-image regression for exploring pixel-level content-relevant prior, we first design a basic unit and then adopt it to construct the de-blocker. Due to different quantization steps, compression artifacts at different spatial scales will appear. Therefore, to capture both global and multi-scale local spatial information, we utilize the non-local operation [62] and dilated convolutions [50] to form the basic unit. Specifically, in each basic unit, we first deploy the non-local operation to capture global spatial information. Similar with [62], the non-local operation is performed as:

$$
M_{out} = M_{in} + \theta(M_{in}) \nu(M_{in})^\dagger \xi(M_{in}) W,
$$

where $M_{in}$ and $M_{out}$ are the input and output features; $\theta(\cdot)$, $\nu(\cdot)$ and $\xi(\cdot)$ are $1 \times 1$ convolutions to reduce the channel number; $\dagger$ is the transpose operation; $W$ is a $1 \times 1$ convolution to perform a hidden-to-output operation. We re-ordered $(\theta \nu \xi)$ to $(\theta \nu \xi)$ according to the associative rule, which can greatly reduce computation complexity by
avoiding large matrix calculation. Then, these global spatial information is sent into three cascaded dilated convolutional layers with different dilation factors. In this way, the basic unit can capture wide-range spatial information and enable a single network to handle multiple quantization steps. Finally, we utilize the basic unit to construct the de-blocker. Note that we adopt dense structures [22] and skip-connections to avoid gradient vanishing and propagate image detail to improve the de-blocking performance.

**Classifier** $f_s(\cdot)$ To further utilize semantic information for image de-blocking, we build a DL-based binary classifier, which receives the image gradients as the input and outputs a single scalar, to represent the probability of being compressed. Specifically, we adopt six standard convolutional layers (with non-linear activations), one global average pooling layer, and two fully-connected layers to construct the classifier. The output scalar is processed by a sigmoid non-linear function to facilitate the binary prediction. Note that the global average pooling operation can convert a feature map into a single scalar, which allows our classifier to handle input gradients of arbitrary sizes. Moreover, since the entire classifier is differentiable, it can participate in the calculation of Equation (9) to update $g$.

As shown in Figure 2, by plugging the de-blocker and classifier into the optimization, the deep unfolding network can be constructed. It should be indicated that our network has a good interpretability, i.e., each network module corresponds to each step in the optimization. The de-blocker accomplishes the exploration of image content-relevant prior to achieve the function of Equation (8), which is to remove JPEG artifacts. The classifier accomplishes the exploration of image gradient-relevant prior and participates in the calculation of Equation (9) so that the gradients can be updated along the direction of ‘clarity’. By taking both pixel-level and semantic-level information into consideration, the de-blocked image can be obtained by Equation (10).

### 3.4. Implementation details

Since our deep unfolding network contains a large number of parameters, it is impractical to manually determine these parameters. Therefore, we make all the learnable parameters be automatically learned from the training samples. We enforce the de-blocker and the classifier to share their own parameters to reduce the number of parameters and thus avoid over-fitting. For the weights in Equations (8) to (10), we let them be discriminatively learned.

In this paper, we adopt a two-stage training strategy to train our deep unfolding network. In the first stage, we only train the classifier via the binary cross entropy loss function:

$$L_{CE} = -\frac{1}{N} \sum_{i=1}^{N} \hat{z}_i \log(z_i) + (1 - \hat{z}_i) \log(1 - z_i),$$  \hspace{1cm} (12)$$

where $N$ is the number of training samples, $z_i$ is the output scalar of the classifier, and $\hat{z_i}$ is the label. We set $\hat{z}_i = 1$ for compressed images and $\hat{z}_i = 0$ for clear images. In the second stage, the parameters of the trained classifier are frozen, and the remaining parameters in the network are trained by using mean absolute error (MAE).

In our deep unfolding network, all convolutional kernel sizes are set as $3 \times 3$. The number of feature maps of de-blocker and classifier are 112 and 32, respectively. We set $K = 5$ and $J = 3$, and use the classic ReLU [25] as the nonlinear activation. The dilation factors are set as 1, 3 and...
5. To train the network, we use the Matlab JPEG encoder to generate JPEG compressed images. The JPEG quality factors (QF) are set to 10, 20 and 30. We use both the training and testing sets from BSD500 [2] as our training set. The training process is conducted on the Y channel image of Y-CrCb space. We randomly generate 64 × 64 training patch pairs with a batch size of 10. We adopt the Adam solver [24] as the optimizer, and the learning rate is fixed to 10⁻⁵. We use TensorFlow [1] to implement our network. Note that we only train one single model to handle all the JPEG compressed images. The JPEG quality factor varies from 20 to 90. We first report the comparison results on the three widely used synthetic datasets, i.e., 5 images in Classic5 [51], 29 images in LIVE1 [39] and 100 images in the validation set of BSD500 [2]. We adopt the PSNR, SSIM [44], and PSNR-B [47] for quantitative evaluations. Since PSNR-B is more sensitive to blocking artifacts than PSNR and S-SIM, it is recommended [8] for use in this de-blocking problem. Table 1 reports the quantitative results and our network achieves the best overall results on all synthetic datasets. Particularly, for the Classic5 dataset with QF = 10, the average gains of our method over the recently proposed RD-N [63] are respectively 0.23dB in PSNR, 0.0209 in SSIM, and 0.46dB in PSNR-B. When compared with the other approaches, our method is far ahead. Note that we only train one model to cover all three QFs, which substantiates the flexibility and effectiveness of our method, in diverse JPEG artifacts contained in these datasets.

In Figure 3, we show two visual comparisons and it is clear that other compared methods can effectively remove most compression artifacts but fail to recover image details. While our network can finely recover the image textures with a better visual quality. This is because our method learns the dual priors, which enables the network to simultaneously utilize the low-level pixel information and high-level semantic information. In this way, not only the image

---

Table 1: PSNR | SSIM | PSNR-B values and parameter numbers comparisons. The best and the second best results are boldfaced and underlined. Our network achieves the best overall results with tolerable parameter numbers.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Classic5</td>
<td>10</td>
<td>28.88</td>
<td>0.8071</td>
<td>28.16</td>
<td>28.39</td>
<td>0.7997</td>
<td>27.59</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>30.92</td>
<td>0.8663</td>
<td>29.75</td>
<td>30.30</td>
<td>0.8584</td>
<td>29.37</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>32.14</td>
<td>0.8914</td>
<td>30.83</td>
<td>31.47</td>
<td>0.8830</td>
<td>30.17</td>
</tr>
<tr>
<td>LIVE1</td>
<td>10</td>
<td>28.65</td>
<td>0.8893</td>
<td>28.26</td>
<td>28.05</td>
<td>0.8252</td>
<td>27.68</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>30.81</td>
<td>0.8781</td>
<td>29.82</td>
<td>30.19</td>
<td>0.8715</td>
<td>29.64</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>32.08</td>
<td>0.9078</td>
<td>30.92</td>
<td>29.41</td>
<td>0.8960</td>
<td>29.36</td>
</tr>
<tr>
<td>BSD500</td>
<td>10</td>
<td>28.23</td>
<td>0.7878</td>
<td>27.38</td>
<td>28.03</td>
<td>0.7824</td>
<td>27.29</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>30.09</td>
<td>0.8510</td>
<td>28.61</td>
<td>29.82</td>
<td>0.8514</td>
<td>28.43</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>31.21</td>
<td>0.8838</td>
<td>29.34</td>
<td>30.87</td>
<td>0.8719</td>
<td>29.15</td>
</tr>
<tr>
<td>Twitter</td>
<td>20</td>
<td>27.61</td>
<td>0.7281</td>
<td>27.53</td>
<td>27.58</td>
<td>0.7274</td>
<td>27.49</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>27.61</td>
<td>0.7281</td>
<td>27.53</td>
<td>27.58</td>
<td>0.7274</td>
<td>27.49</td>
</tr>
</tbody>
</table>

# Params (×10⁴) 1.06 0.21 6.69

4. Experimental results

We compare our network with three model-driven methods: Shape-Adaptive DCT (SADCT) [12], Layer Decomposition (LD) [27] and PCA basis Learning (PCA) [40], and several deep learning-based methods: Artifacts Reduction Convolutional Neural Network (ARCNN) [8], Trainable Nonlinear Reaction Diffusion (TNRD) [6], Denoising Convolutional Neural Network (DnCNN) [57], Learning Parameterized Image Operators (LPIO) [11], Memory Network (M-Net) [42], Deep Convolutional Sparse Coding (DCSC) [13], Residual Non-local Attention Networks (R-NAN) [62] and Residual Dense Network (RDN) [63].

4.1. Comparisons on synthetic datasets

We first report the comparison results on the three widely used synthetic datasets, i.e., 5 images in Classic5 [51], 29 images in LIVE1 [39] and 100 images in the validation set of BSD500 [2]. We adopt the PSNR, SSIM [44], and PSNR-B [47] for quantitative evaluations. Since PSNR-B is more sensitive to blocking artifacts than PSNR and S-SIM, it is recommended [8] for use in this de-blocking problem. Table 1 reports the quantitative results and our network achieves the best overall results on all synthetic datasets. Particularly, for the Classic5 dataset with QF = 10, the average gains of our method over the recently proposed RD-N [63] are respectively 0.23dB in PSNR, 0.0209 in SSIM, and 0.46dB in PSNR-B. When compared with the other approaches, our method is far ahead. Note that we only train one model to cover all three QFs, which substantiates the flexibility and effectiveness of our method, in diverse JPEG artifacts contained in these datasets.
content-related compression artifacts are removed, but also the image gradient-related clarity is recovered.

4.2. Comparisons on real-world use cases

Online social media softwares have been widely used for message publishing and sharing. To reduce transmission and storage consumption, these platforms usually compresses and re-scales the original images on the server-side. This leads to undesired and unavoidable compression artifacts appearance when users view the posted images. To complement the existing Twitter dataset [8] and serve the relevant research communities on this issue in real scenarios, we manually construct a new dataset based on the popular social media WeChat. The dataset contains 300 images and their WeChat-compressed versions. To avoid out-of-memory caused by excessive image resolution, we first randomly crop the images and then perform the de-blocking operation and measurement calculation. We show quantitative results in Table 1, in which our model consistently generates the best overall performance due to the effective dual priors. Figure 4 shows two visual comparisons. Due to different compression strategies, it is clear that the two compressed images contain different appearances of artifacts, and our method still achieves a superior performance than other compared methods. It is observed that our method can generate clearer results than other competing ones.

4.3. Analysis

Analysis on classifier $f_s(\cdot)$ We first visualize the derivative of the classifier at different stages in Figure 5. By referring to $\nabla x_0$, the areas of large derivative values, which are normalized for visualization, of g are basically the same as the areas of compression artifacts. Since the gradient represents the increasing direction of the function, using Equation (9) can update $g$ along the direction of less artifacts.

In Figure 6, we visualize the activations of the last convolutional layer in the classifier. It is clear that our classifier can effectively distinguish clear images and their compressed versions generated from different platforms. Since we use the classifier to explore gradient-relevant priors, the differences between compressed and clear activations are mainly concentrated in areas such as image texture and structure, and edges of compression artifacts. This proves that using our gradient-relevant prior can provide the network with constraints related to ‘clarity’.

We also show one visual result in Figure 7 to demonstrate the effect of $f_p(\cdot)$. It is clear that using only pixel-level $f_p(\cdot)$ can obtain a de-blocked result with blurred edges. By adding the semantic-level $f_s(\cdot)$, the edges become more clear with a better visual quality.

Analysis on de-blocker $f_p(\cdot)$ For the pixel-level de-blocker, we compare with the scenarios by using three operators. Specifically, we test the handcrafted $\ell_1$ sparsity [17], the DL-based DnCNN [57], and our default $f_p(\cdot)$. Table 2 shows the quantitative comparisons on BSD500 (QF = 10), and using our $f_p(\cdot)$ achieves the best results. Compared

---

Footnote: 1The dataset can be found at: https://xueyangfu.github.io/
with the other two de-blockers, our $f_p(\cdot)$ can extract both
global and multi-scale local features. This is particularly
suitable for image de-blocking since compression artifacts
usually have the appearance of different spatial scales.

**Analysis on stage $K$** We also analyze the effect of stage
number $K$ and show the quantitative results on *BSD500* (QF = 10) in Figure 8. Using $K = 1$ as a baseline, it is clear
that the performance has an obvious improvement with 3 stages. When $K = 7$, 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 $K = 5$ as the default stage number.

### 5. Conclusion

In this paper, we propose two data-driven priors for
JPEG compression artifacts removal. Specifically, we de-
sign one pixel-level prior and one semantic-level prior to
provide regressive and discriminative information, respec-
tively. We then embed these two priors into a variational
model and develop an alternative optimization algorithm to
solve it. This optimization algorithm is further unfolded
into a deep network, in which the dual priors can be ef-
fectively explored from training samples. Experiments on
benchmarks and real-world use cases show that our method
performs favorably against state-of-the-art methods.

---

**Table 2: Comparisons on using different de-blockers.**

<table>
<thead>
<tr>
<th>Proximal operators</th>
<th>PSNR</th>
<th>SSIM</th>
<th>PNSR-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell_1$ sparsity [17]</td>
<td>28.18</td>
<td>0.7985</td>
<td>27.28</td>
</tr>
<tr>
<td>DnCNN [57]</td>
<td>28.97</td>
<td>0.8094</td>
<td>27.95</td>
</tr>
<tr>
<td>Our $f_p(\cdot)$</td>
<td>29.48</td>
<td>0.8146</td>
<td>29.13</td>
</tr>
</tbody>
</table>

---

**Figure 4:** Visual comparisons on two real-world use cases.

**Figure 6:** Activations of a feature map in the classifier. From top to bottom: *BSD500, Twitter* and *WeChat.*

**Figure 7:** Effectiveness of $f_s(\cdot).$

**Figure 8:** Quantitative results on different stage numbers $K$. 
References


