Supplement to "Early Exit Or Not: Resource-Efficient Blind Quality Enhancement for Compressed Images"

Qunliang Xing$^{1[0000-0002-3007-716X]}$, Mai Xu$^{1,2[0000-0002-0277-3301]}$, Tianyi Li$^{1[0000-0001-7038-7798]}$, and Zhenyu Guan$^{1[0000-0002-3959-338X]}$

^{1} School of Electronic and Information Engineering, Beihang University
^{2} Hangzhou Innovation Institute, Beihang University
{xingql,maixu,tianyili,guanzhenyu}@buaa.edu.cn

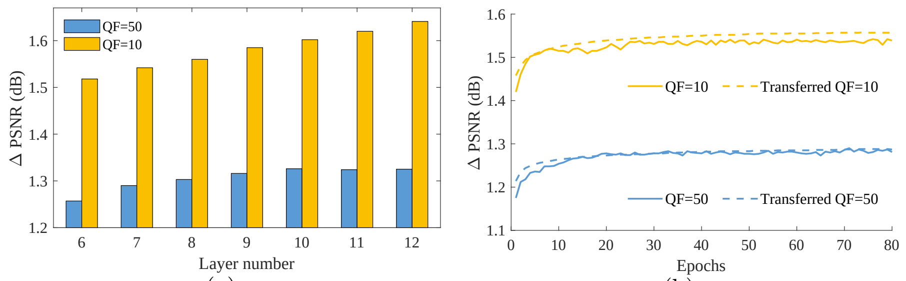
This material provides the motivation of enhancing JPEG-compressed images in Sec. 1, additional details about image quality assessment module in Sec. 2, efficiency performance of enhancing JPEG-compressed images in Sec. 3, and more enhanced examples in Sec. 4.

1 Motivation of Enhancing JPEG-compressed Images

In this section, we conduct experiments on JPEG-compressed images to prove the two propositions of our paper: **Proposition 1**: "Easy" samples (i.e., high-quality compressed images) can be simply enhanced, while "hard" samples (i.e., low-quality compressed images) should be further enhanced; **Proposition 2**: The quality enhancement process with different computational complexity can be jointly optimized in a single network through an "easy to hard" manner, rather than a "hard to easy" manner.

1.1 Proof of Proposition 1

We construct a series of vanilla CNNs with the layer number from 6 to 12. Each layer includes $32\times3\times3$ filters, except for the last layer with $1\times3\times3$ filter. Beside, ReLU [7] activation and global residual learning [2] are adopted. The training, validation and test sets (including 400, 100 and 100 raw images, respectively) are randomly selected from RAISE without overlapping. They are all compressed by the JPEG encoder of Python Imaging Library (PIL) [5] with quality factor (QF) = 50 and 10 for obtaining "easy" and "hard" samples, respectively. We train these vanilla CNNs with the "easy" samples, and then obtain converged models "QF = 50". Similarly, we train the CNNs with the "hard" samples and then obtain converged models "QF = 10". As shown in Fig. 1 (a), the performance of QF = 10 models improves significantly with the increase of layer numbers, while the performance of QF = 50 models gradually becomes saturated once the layer number excesses 10. Therefore, it is possible to enhance the "easy" samples with a simpler architecture and fewer computational resources, and to further enhance the "hard" samples in a more elaborate process.



(a) (b)

Fig. 1. (a) Average ΔPSNR (dB) of vanilla CNNs over the test set. (b) Average ΔPSNR (dB) curves alongside increased epochs, for vanilla CNN models and their transferred models over the validation set during the training stage.

1.2 Proof of Proposition 2

Here, we investigate the efficacy of "easy to hard" strategy on image quality enhancement through the experiments of transfer learning. If the filters learned from "easy" samples can be transferred to enhance "hard" samples more successfully than the opposite manner, then our proposition can be proved. Here, we construct 2 identical vanilla CNNs with 7 convolutional layers. The other settings conform to the above. We train these 2 models with the training sets of images compressed at QF = 50 and 10, respectively, and accordingly these 2 models are called "QF = 50" and "QF = 10". After convergence, they exchange their parameters for the first 2 layers and restart training with their own training sets. Note that the exchanged parameters are frozen during the training stage. We name the model transferred from QF = 10 to QF = 50 as "transferred QF = 50" and the model transferred from QF = 50 to QF = 10 as "transferred QF = 10". Fig. 1 (b) shows the validation-epoch curves of the original 2 models and their transferred models. As shown in this figure, the transferred QF = 10 model improves the performance of the QF = 10 model, while the transferred QF = 50 model does not benefit the performance of the QF = 50 model. Consequently, the joint simple and elaborate enhancement process should be conducted in an "easy to hard" manner rather than a "hard to easy" manner. In summary, proposition 2 can be proved.

2 Details about image quality assessment module

In this section, we present details about the mechanism of Image Quality Assessment Module (IQAM).

2.1 Image Partition

We first partition the image into patches. The obtained patches should cover all the potential block boundaries, as shown in Fig. 2. For a JPEG-compressed

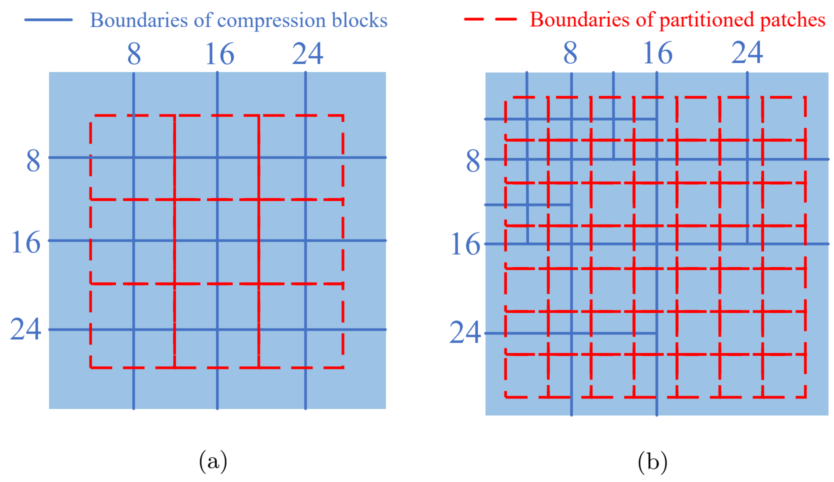

Boundaries of compression blocks
Boundaries of partitioned patches
8
16
24
8
16
24
8
16
24
8
16
24
(a)
(b)

Fig. 2. Examples of compression blocks (with blue solid boundaries) and obtained patches (with red dotted boundaries) of a JPEG-compressed image (a) and an HEVC-MSP-compressed image (b).

image, it is split into 8×8 blocks from the top-left. Therefore, 8×8 patches are obtained to cover all block boundaries, as depicted in Fig. 2 (a). For an HEVC-MSP-compressed image, it is split into coding units (CUs), transform units (TUs) and prediction units (PUs) in a tree structure [8]. Among these units, CUs and TUs may contribute to blocky effects, and their minimum size is 4×4. Therefore, we partition the HEVC-MSP-compressed image into 4×4 patches, as depicted in Fig. 2 (b).

2.2 Patches Classification

We take 4×4 patches as an example. We classify these patches into smooth ones and textured ones, according to their sum of squared non-DC Tchebichef moment (SSTM) values [6, 3]:

$$\mathbf{M} = \begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{pmatrix}, \tag{1}$$

$$\text{SSTM} = \left(\sum_{i=0}^{3} \sum_{j=0}^{3} m_{ij}^2 \right) - m_{00}^2, \tag{2}$$

where $\mathbf{M}$ denotes the Tchebichef moment of each patch. In fact, SSTM can be used for measuring block energy, and it is higher for textured patches and lower for smooth patches. Therefore, patches can be classified by comparing SSTM with a threshold T_{SSTM}. If SSTM $< T_{\mathrm{SSTM}}$, the patch is classified as a smooth one. Otherwise, it will be classified as a textured one.

2.3 Evaluation on Blocky Effects

For smooth patches, blocky effects are the dominant factor that degrades the quality. Therefore, we evaluate the score of blocky effects in the textured patches. Motivated by [4], the ratio of the summation of absolute 3th order moment values to the summation of absolute non-DC moment values can reflect the energy of vertical and horizontal blocky effects. Specifically, the energy of vertical and horizontal blocky effects, denoted by e_{v} and e_{h}, can be reflected by the following two metrics respectively:

$$e_{\mathrm{h}} = \frac{\sum_{i=0}^{3} |m_{i3}|}{\left(\sum_{i=0}^{3}\sum_{j=0}^{3} |m_{ij}|\right) - |m_{00}| + C}, \tag{3}$$

$$e_{\mathrm{v}} = \frac{\sum_{j=0}^{3} |m_{3j}|}{\left(\sum_{i=0}^{3}\sum_{j=0}^{3} |m_{ij}|\right) - |m_{00}| + C}, \tag{4}$$

where C is a small constant to ensure the numerical stability. The bigger $e_{\mathrm{v}}/e_{\mathrm{h}}$, the slighter the vertical/horizontal blocky effects. When $e_{\mathrm{v}}/e_{\mathrm{h}}$ is bigger than a threshold T_e, the blocky effects are too slight to notice:

$$e_{\mathrm{v}} = \begin{cases} e_{\mathrm{v}} & e_{\mathrm{v}} < T_e \\ T_e & e_{\mathrm{v}} \geq T_e, \end{cases} \tag{5}$$

$$e_{\mathrm{h}} = \begin{cases} e_{\mathrm{h}} & e_{\mathrm{h}} < T_e \\ T_e & e_{\mathrm{h}} \geq T_e, \end{cases} \tag{6}$$

Finally, the quality score of the smooth patch is calculated as:

$$\mathcal{Q}_{\mathrm{S}} = \log_{(1-T_e)}\left(1 - \frac{e_{\mathrm{v}} + e_{\mathrm{h}}}{2}\right). \tag{7}$$

Note that a higher $\mathcal{Q}_{\mathrm{S}}$ indicates better quality of the smooth patch.

2.4 Evaluation on Blurring

For textured patches, blurring is the dominant factor that degrades the quality. Therefore, we evaluate the score of blurring in the textured patches. Each textured patch is blurred by a Gaussian filter:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2}\exp\left(\frac{-(x^2+y^2)}{2\sigma^2}\right), \tag{8}$$

where σ is the stand deviation of the Gaussian filter; (x, y) denotes the coordinate of each image pixel. Following [3], the filter size is set to 3×3. The standard deviation is set to 5. Then, we can obtain the Tchebichef moments of the blurred patch:

$$\mathbf{M}' = \begin{pmatrix} m'_{00} & m'_{01} & m'_{02} & m'_{03} \\ m'_{10} & m'_{11} & m'_{12} & m'_{13} \\ m'_{20} & m'_{21} & m'_{22} & m'_{23} \\ m'_{30} & m'_{31} & m'_{32} & m'_{33} \end{pmatrix}. \tag{9}$$

The similarity of $\mathbf{M}$ and $\mathbf{M}'$ is calculated as:

$$\mathbf{S}(i,j) = \frac{2m_{ij}m'_{ij} + C}{(m_{ij})^2 + (m'_{ij})^2 + C}, \ i,j = 0,1,2,3. \tag{10}$$

The similarity value of severely blurred patch is bigger than that of slightly blurred patch. Therefore, we compute the quality score of textured patch as follows,

$$\mathcal{Q}_{\mathrm{T}} = 1 - \frac{1}{3 \times 3} \sum_{i=0}^{3} \sum_{j=0}^{3} \mathbf{S}(i,j). \tag{11}$$

Note that a higher $\mathcal{Q}_{\mathrm{T}}$ indicates better quality of the textured patch. Also note that this formulation of quality score of textured patch is different from that in [3], which directly takes the average similarity values as the quality score. It is inconsistent with the quality score of smooth patch, since the latter increases when the quality is better.

2.5 Quality Score

Finally, we calculate the average quality scores of all smooth patches:

$$\bar{\mathcal{Q}}_{\mathrm{S}} = \sum_{k=1}^{N_{\mathrm{S}}} \mathcal{Q}_{\mathrm{S,k}}, \tag{12}$$

where $\mathcal{Q}_{\mathrm{S,k}}$ is the k-th smooth patch and N_{S} is the number of smooth patches. Similarly, we can obtain the average quality scores of all textured patches $\bar{\mathcal{Q}}_{\mathrm{T}}$. The final quality score of the image can be generated as:

$$\mathcal{Q} = (\bar{\mathcal{Q}}_{\mathrm{S}})^{\alpha} \cdot (\bar{\mathcal{Q}}_{\mathrm{T}})^{\beta}, \tag{13}$$

where $\alpha > \beta$. It is because blocky effects are the dominant distortion [3], especially for low bit-rate compression.

2.6 Implementation

We set $T_{\mathrm{SSTM}} = 4e{-}3$, $C = 1e{-}8$, $T_e = 5e{-}2$ and $\alpha = 0.9 > \beta = 0.1$ through experiments on a validation set (including 1000 pairs of raw/compressed images randomly selected from RAISE [1]) compressed at 5 different QPs (QP $= 22, 27, 32, 37, 42$).

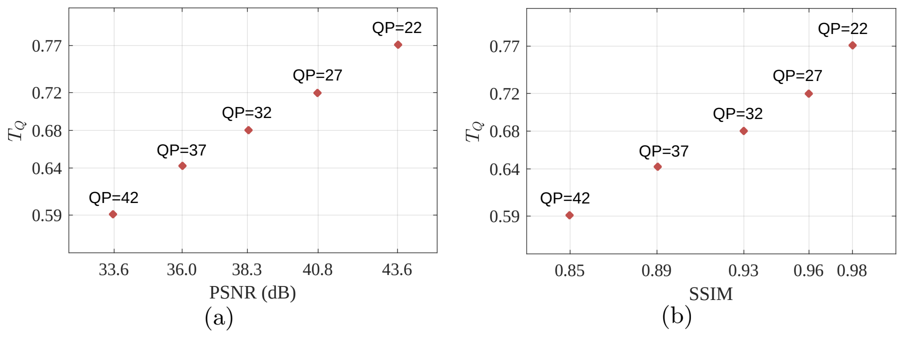


Fig. 3. (a) Average PSNR (dB) and T_Q of the HEVC validation set compressed at 5 QPs. (b) Average SSIM and T_Q of the HEVC validation set compressed at 5 QPs.

2.7 Relation between T_Q and Objective Quality Metrics

The generated quality score T_Q is highly and positively correlated to objective quality metrics, such as peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) index. To verify this, we calculate the average T_Q of the validation set. As shown in Fig. 3 (a) and (b), the PSNR, SSIM and T_Q increase along with the decreased QP values. Therefore, the strong and positive correlation between T_Q and objective quality metrics PSNR and SSIM can be verified.

3 Efficiency of Enhancing JPEG-compressed Images

We also validate the efficiency of the RBQE approach when enhancing the quality of JPEG-compressed images in terms of the average consumed FLOPs. Our RBQE approach consumes only 26.9 GMacs for the "hardest" samples, i.e., the images compressed at QF = 10. In contrast, DCAD, QE-CNN, CBDNet and DnCNN consume constantly 77.8, 118.4, 160.5 and 175.8 GMacs for all samples that are either "easy" or "hard" samples compressed at 5 different QFs. Therefore, our RBQE approach is much more efficient than compared approaches when enhancing the quality of JPEG-compressed images.

4 Enhanced Samples

Fig. 4 visualizes the enhanced samples by our RBQE and other compared approaches. In particular, the smooth background can be finely restored by our RBQE approach, while other approaches are ineffective to suppress compression artifacts surrounding the lines, clothes and petals.

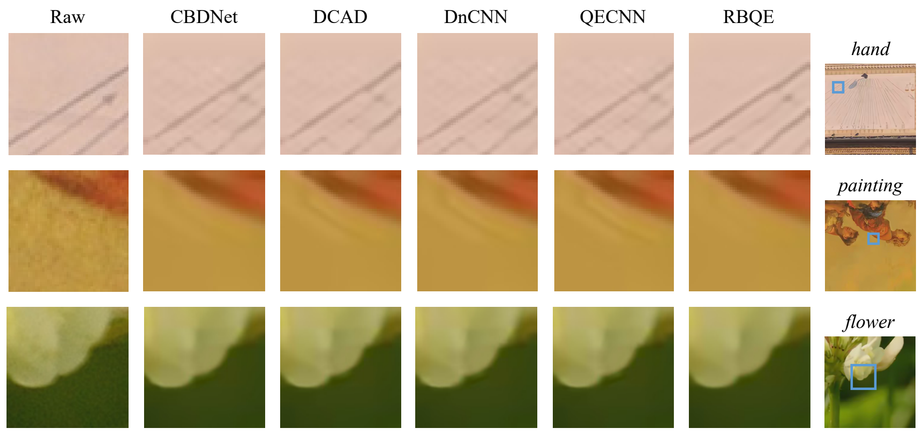


Fig. 4. Enhanced test samples. We observe significant suppression by RBQE of compression artifacts surrounding the lines, clothes and petals.

References

1. Dang-Nguyen, D.T., Pasquini, C., Conotter, V., Boato, G.: Raise: A raw images dataset for digital image forensics. In: The 6th ACM Multimedia Systems Conference. pp. 219–224. ACM (2015)
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016)
3. Li, L., Zhou, Y., Lin, W., Wu, J., Zhang, X., Chen, B.: No-reference quality assessment of deblocked images. Neurocomputing **177**, 572–584 (2016)
4. Li, L., Zhu, H., Yang, G., Qian, J.: Referenceless measure of blocking artifacts by Tchebichef kernel analysis. IEEE Signal Processing Letters **21**(1), 122–125 (2013)
5. Lundh, F.: Python imaging library (PIL), http://www.pythonware.com/products/pil
6. Mukundan, R., Ong, S., Lee, P.A.: Image analysis by Tchebichef moments. IEEE Transactions on Image Processing (TIP) **10**(9), 1357–1364 (2001)
7. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: The 27th International Conference on Machine Learning (ICML). pp. 807–814 (2010)
8. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (HEVC) standard. IEEE Transactions on Circuits and Systems for Video Technology (TCSVT) **22**(12), 1649–1668 (2012)