Learned Compression Artifact Removal by Deep Residual Networks

Ogun Kirmemis Gonca Bakar A. Murat Tekalp
Department of Electrical and Electronics Engineering, Koc University, 34450 Istanbul, Turkey
{okirmemis16,gbakar15,mtekalp}@ku.edu.tr

Abstract

We propose a method for learned compression artifact removal by post-processing of BPG compressed images. We trained three networks of different sizes. We encoded input images using BPG with different QP values. We submitted the best combination of test images, encoded with different QP and post-processed by one of three networks, which satisfy the file size and decode time constraints imposed by the Challenge. The selection of the best combination is posed as an integer programming problem. Although the visual improvements in image quality is impressive, the average PSNR improvement for the results is about 0.5 dB.

1. Introduction

The mainstream approach for lossy image compression since 1980's has been transform coding, using discrete cosine transform (DCT) or discrete wavelet transform (DWT) for data decorrelation followed by uniform quantization and entropy coding. The JPEG standard using the DCT has been the most successful and widely deployed lossy image compression technology. JPEG2000, which uses the DWT, is the technology used by the motion picture industry for frame by frame compression of movies.

Recently, the state of the art in lossy image coding has shifted to the better portable graphics (BPG) codec [1], which is also a transform coder derived from intra-frame coding tools in the high-efficiency video coding (HEVC) video coder. The royalty-free WebP codec, which is derived from the intra-frame coding tools of the VP9 video coder, also outperforms JPEG but is slightly inferior to BPG.

With the advent of deep learning, which led to significant achievements in computer vision, there is growing interest in applying end-to-end deep learning to image compression. Many works have already been published on novel encoder/decoder architectures, learned transforms, and learning to better quantize real variables [20] [16] [2] [6] [21] [9] [3] [13] [17].

Our hypothesis in this paper is that end-to-end learned image compression methods have not yet matured to the level to beat the state of the art signal-processing-based transform codecs, e.g., the BPG codec. Hence, we propose a learned post-processing method to improve the visual perceptual quality of BPG compressed images.

2. Related Works

Available post-processing methods can be classified as traditional filters and learned artifact reduction methods.

Traditional filters for removal of compression artifacts include deblocking and deringing filters that were proposed as in-loop or post-processing filters to be used with image/video compression standards. An example for in-loop filters is the HEVC deblocking filter [15]. Commonly used post-processing filters are those of Foi *et al.* [7], which proposed thresholding in shape-adaptive DCT domain for deblocking; Zhang *et al.* [22], which proposed similarity priors for image blocks to reduce compression artifacts by estimating the transform coefficients of overlapped blocks from non-local blocks; and Dar *et al.* [4], which modeled the compression-decompression procedure as a linear system and then estimate the solution to the inverse problem.

Methods using deep learning for post-processing of compression artifacts include Dong *et al.* [5], which proposes 4 layer convolutional neural network for deblocking and deblurring of compressed images; Svoboda *et al.* [18], which proposes an 8 layer residual network and add a loss term defined by the difference between the first partial derivatives of the target and output images to the MSE loss; and Galteri *et al.* [8], which proposes a solution based on generative adversarial networks (GAN) to reduce compression artifacts.

3. System and Network Architecture

The proposed system is depicted in Figure 1. The encoder unit uses the BPG encoder [1]. The decoder unit comprises of a BGP decoder and post-processing network. Since we trained three different networks, the encoder adds a byte in the beginning of the bit-stream to signal the choice of the neural network. Decoder unit reads the first byte and sends the rest of the bitstream to the BPG decoder. Then, the decompressed image is processed by the selected post-processing network yielding the final output image.

The proposed neural network is a modified version of the enhanced deep super-resolution (EDSR) network [14], which is based on SRResNet [12] architecture. The main difference between EDSR and ours is that we use SELU activation function [11] instead of ReLU as shown in Figure 3, since SELU activation enables faster learning [11]. We also remove the upsampling blocks of SRResNet. Un-

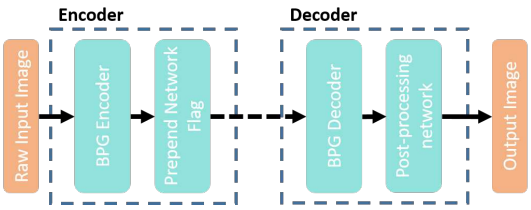


Figure 1: Block diagram of encoding/decoding system.

like the networks in [14] and [12], we add a direct shortcut connection from the input RGB image to output RGB image. Since our aim is to restore compressed images, the input image is closer to the output image than the randomly initialized network from the point of optimization. Because of this, we also multiply the contribution of the network with 0.1. This way the overall function for the network is closer to identity function so that the predictions of the network resemble the input image at the beginning of training.

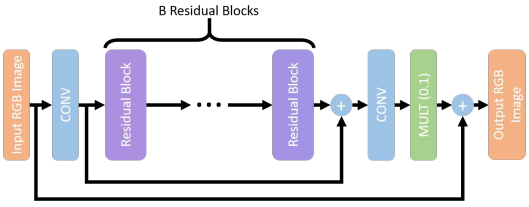


Figure 2: Architecture of the proposed post-processing network with *B* residual blocks. There is a direct shortcut connection from the input image to output image.

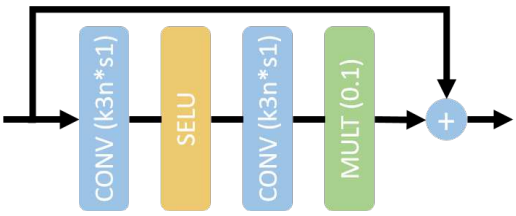


Figure 3: A residual block with kernel size *k*, number of feature maps *n* and stride *s*. We also employ residual scaling to make training easier as suggested in [19].

In order to comply with RAM requirements (8 GB) on the evaluation server, our decoder divides the input image to 4 blocks and processes these blocks separately. We employ an Overlap-Save method, to produce the same output as if the whole image is processed at once. In order to apply the Overlap-Save method, the effective kernel size of the neural network has to be calculated. For a network which has l convolutional layers with kernel size of k, the effective kernel size E of the overall network is $E = (k-1)l + 1$. After we divide the input image to 4 blocks, we pad each block on all sides to size $\frac{E+1}{2}$. Then, we pass these blocks through the network. When merging the output blocks, we discard overlapping pixels and construct the output image.

4. Training Method

We train 3 models with different depth B, referring to the number of blocks, and width n, referring to the number of feature maps in Figures 2 and 3. These networks are called MVGL A (B=32, n=96) with 5.40M parameters, MVGL B (B=8, n=96) with 1.42M parameters, and MVGL C (B=32, n=48) with 1.35M parameters. MVGL A is trained with batch size of 24, while both MVGL B and MVGL C are trained with batch size of 64.

We train all networks with the given training set consisting of 1633 images. We encode the training images using the BPG encoder with QP=40 at the highest compression level (9). QP=40 is the minimum QP value that we can choose to fit the validation set into the given constraint of 0.15 bits per pixel (bpp).

We calculate the mean of the RGB channels of all images in the training set (single mean per channel for the training set), and subtract them from both target images and their compressed/decompressed versions before feeding them into the network. We train networks on MSE loss using the Adam optimizer[10] with the default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$). The learning rate is initialized to 0.001 and is halved at every 500^{th} epoch. Networks are trained on 96×96 random crops without any data augmentation. A random patch of size 96×96 is cropped randomly from every training image to create training batch for an epoch. We stop training networks upon convergence, that is, when there is no improvement for 50 epochs.

5. Evaluation

We present PSNR and MS-SSIM results for different QP and networks on the given training, validation, and test sets.

5.1. Results on Training and Validation Sets

The average PSNR and MS-SSIM results on the training and validation sets encoded with QP=40 are shown in Table 1. MVGL A is the best performing network with PSNR gain of $\approx$ 0.7dB on both training and validation sets, since it has the largest number of parameters (weights). MVGL B and MVGL C networks give comparable results with $\approx$ 0.3-0.4 dB PSNR improvements, since the number of parameters in these networks are close to each other.

Table 1: Results on the training set (0.169 bpp) and validation set (0.149 bpp) where QP=40 for BPG compression.

	Training Set		Validation Set	
Method	PSNR	MS-SSIM	PSNR	MS-SSIM
BPG	30.529	0.948	30.842	0.948
MVGL A	31.221	0.955	31.533	0.955
MVGL B	30.899	0.951	31.223	0.952
MVGL C	30.952	0.952	31.277	0.950

5.2. Encoding the Test Set with File Size Constraint

Suppose there are N images in the test set and we need to choose the best QP value of out of M different values for each image to maximize the average PSNR of BPG encoding subject to a file size constraint. We formulate this problem as an integer linear programming problem, given by

$$\min_{x_i} \quad \sum_{i=1}^{N} f_i^T x_i \tag{1a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} b_i^T x_i \leq FileSizeLimit, \tag{1b}$$

$$\mathbf{1}_{1\times M} x_i = 1,\ \forall i = 1, 2, \cdots N, \tag{1c}$$

$$x_{i_j} \in \{0, 1\},\ \forall i = 1, 2, \cdots N,\ \forall j = 1, 2, \cdots M \tag{1d}$$

where x_i is $M \times 1$ one-hot vector such that the entry which equals 1 indicates the QP selected for the i^{th} image, f_i is $M \times 1$ vector whose components are the sum of squared error between the raw and encoded images for different QPs, and b_i is $M \times 1$ vector whose components denote the file size when i^{th} image is encoded with all possible QP. Eqn. 1b enforces that the sum of sizes of all images are below the given file size constraint $FileSizeLimit$. Constraints 1c and 1d require that only one QP is selected for each image.

We solved this problem for $N = 286$ and $M = 5$ corresponding to QP values 38 to 42. Solution of this problem reveals that we should encode 1 image with QP=38, 109 images with QP=40, 120 images with QP=41, and 56 images with QP=42 so that the average bitrate is 0.15 bpp.

5.3. Results on the Test Set

We encoded $N = 286$ images in the test set with the QP values determined above to meet the file size constraint. We now need to determine which of the three networks to use for post-processing of each image. Our results in Table 2 indicate that the best results can be obtained by the network MVGL A; however, the total processing time was too long. MVGL B and MVGL C are considerably faster but yield lower PSNR improvements.

Submitted Results: Because the training of MVGL C was not complete by the Challenge submission deadline, we decided to process 71 images by MVGL A and the remaining images by MVGL B so that the total processing time is less than $\approx$ 45 hours on Intel i7-3630QM 2.40GHz CPU. This method is called MVGL in Table 2 and is our submission to the Challenge. 71 images to be processed by MVGL A are selected such that they yield the biggest PSNR improvement when processed by MVGL A instead of MVGL B. Had we considered combination of MVGL A and MVGL C for our submission, the average PSNR would be 30.180.

Complete Results: Table 3 presents average PSNR and MS-SSIM values for all combinations of encoding all images in the test set with QP values 39-43 and post-processing them by all three networks. In each row of the table, we encode all images in the test set with the same QP. Table 3 shows that all three networks provide solid PSNR gains across different QP values which means that the networks generalize for different QP values well even though they are only trained with images encoded using QP=40. All images show impressive visual quality improvement. Two example visual results are shown in Figure 4.

Table 2: Average PSNR and MS-SSIM and time (mins) for the Test Set. The average bit-rate is 0.15 bpp. PSNR gain is the difference between PSNR of post-processed and BPG.

Method	PSNR	PSNR Gain	MS-SSIM	Time
BPG	29.692	-	0.944	
MVGL A	30.267	0.575	0.950	6210
MVGL B	30.011	0.319	0.947	1582
MVGL C	30.052	0.360	0.947	634
MVGL	30.135	0.443	0.948	2725

6. Conclusions

The success of the proposed deep learning methods for post-processing of compressed/decompressed images depends on availability of sufficient processing power for both training and testing. Our results (comparing Network B and Network C) show that the average PSNR over the test set (for the same rate) improves by the depth of the network. However, the computational load of test phase with even moderately deep networks can be demanding. As a result, we were not able to submit our best results for the challenge, but only those results that conform with the computational constraints imposed by the Challenge administrators.

References

[1] Better Portable Graphics encoder/decoder and bitstream specification. `https://bellard.org/bpg/`, `https://bellard.org/bpg/bpg_spec.txt`.

[2] J. Balle, V. Laparra, and E. P. Simoncelli. End-to-end optimized image compression. In *ICLR*, April 2017.

[3] M. Covell, N. Johnston, D. Minnen, S. Hwang, J. Shor, S. Singh, D. Vincent, and G. Toderici. Target-quality image compression with recurrent, convolutional neural networks. In *eprint arXiv:abs/1705.06687*, May 2017.

[4] Y. Dar, A. M. Bruckstein, M. Elad, and R. Giryes. Postprocessing of compressed images via sequential denoising. In *eprint arXiv:abs/1510.09041v2*, Mar. 2016.

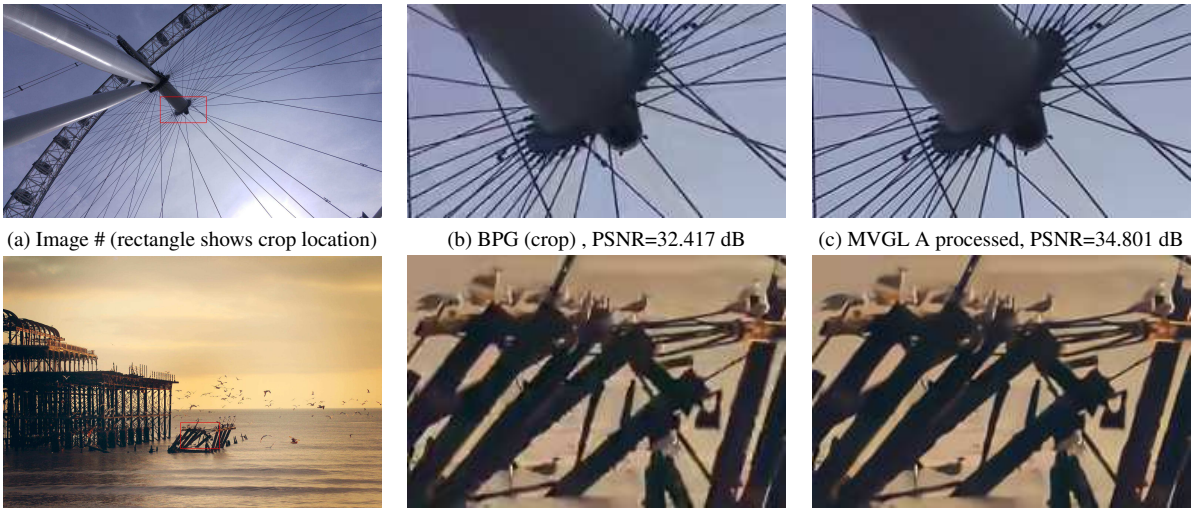
[5] C. Dong, Y. Deng, C. C. Loy, and X. Tang. Compression artifacts reduction by a deep convolutional network. In *Proc. of Int. Conf. on Computer Vision (ICCV)*, 2015.

[6] T. Dumas, A. Roumy, and C. Guillemot. Image compression with stochastic winner-take-all auto-encoder. In *IEEE ICASSP*, Mar 2017.

[7] A. Foi, V. Katkovnik, and K. Egiazarian. Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images. *IEEE Trans. on Image Processing*, 16(5):1395–1411, May 2007.

Table 3: Results for the test set encoded with different QP values.

	BPG		MVGL A		MVGL B		MVGL C		
QP	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	Bitrate (bpp)
39	30.833	0.954	31.486	0.960	31.189	0.957	31.238	0.957	0.206
40	30.333	0.949	30.956	0.955	30.674	0.952	30.720	0.953	0.179
41	29.735	0.943	30.334	0.950	30.064	0.946	30.108	0.947	0.152
42	29.249	0.938	29.802	0.944	29.554	0.941	29.594	0.941	0.132
43	28.687	0.930	29.200	0.937	28.970	0.934	29.008	0.934	0.111



(a) Image # (rectangle shows crop location)
(b) BPG (crop) , PSNR=32.417 dB
(c) MVGL A processed, PSNR=34.801 dB
(d) Image # (rectangle shows crop location)
(e) BPG (crop), PSNR=31.798 dB
(f) MVGL A processed, PSNR=33.445 dB

Figure 4: Visual results for two images from the Test Set.

[8] L. Galteri, L. Seidenari, M. Bertini, and A. Del Bimbo. Deep generative adversarial compression artifact removal. In *eprint arXiv:abs/1704.02518*, Dec. 2017.

[9] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. J. Hwang, J. Shor, and G. Toderici. Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. In *eprint arXiv:abs/1703.10114*, Mar 2017.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd Int. Conf. on Learning Representations (ICLR)*, May 2015.

[11] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In *eprint arXiv:abs/1706.02515*, Sept. 2017.

[12] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *eprint arXiv:abs/1609.04802*, May 2017.

[13] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang. Target-quality image compression with recurrent, convolutional neural networks. In *eprint arXiv:abs/1703.10553*, Mar 2017.

[14] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[15] A. Norkin, G. Bjontegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, M. Zhou, and G. V. der Auwera. Hevc deblocking filter. *IEEE Trans. on Circuits and Systems for Video Tech.*, 22(12):1746–1754, Dec 2012.

[16] O. Rippel and L. Bourdev. Real-time adaptive image compression. In *ICML*, May 2017.

[17] S. Santurkar, D. Budden, and N. Shavit. Generative compression. In *eprint arXiv:abs/1703.01467*, Jun 2017.

[18] P. Svoboda, M. Hradis, D. Barina, and P. Zemcik. Compression artifacts removal using convolutional neural networks. *Journal of WSCG*, 24(2):63–72, 2016.

[19] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4,inception-resnet and the impact of residual connections on learning. In *eprint arXiv:abs/1602.07261*, Aug. 2016.

[20] L. Theis, W. Shi, A. Cunningham, , and F. Huszar. Lossy image compression with compressive autoencoders. In *ICLR*, April 2017.

[21] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell. Full resolution image compression with recurrent neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5435–5443, July 2017.

[22] 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. on Image Processing*, 22(12):4613–4626, Dec 2013.