Decoder Side Image Quality Enhancement exploiting Inter-channel Correlation in a 3-stage CNN: Submission to CLIC 2018

Kai Cui and Eckehard Steinbach
Chair of Media Technology, Technical University of Munich
Munich, Germany
kai.cui@tum.de, eckehard.steinbach@tum.de

Abstract

In this paper, we describe our submission to the workshop and challenge on learned image compression (CLIC) hosted at CVPR 2018. Lossy compressed images usually suffer from unpleasant artifacts, especially when the bit-rate is low. In order to improve the image quality without spending extra bit-rate, decoder side quality enhancement becomes necessary. Most approaches focus on spatial information exploration, in which the quality enhancement is usually only performed on the luminance component or the gray-scale images which makes the inter-channel correlation is neglected. Motivated by the characteristics of compressed images, a 3-stage CNN based approach is proposed in this paper, which can exploit most of the inter-channel correlation to enhance the image quality at the decoder side. Both objective and subjective evaluations show the noticeable quality improvements compared to Better Portable Graphics (BPG), the state-of-the-art image codec.

1. Introduction

Most of the modern lossy image and video codecs (e.g. JPEG, BPG[1], H.264, HEVC) are block based. The compressed images and videos often suffer from visible distortion (e.g. block and ringing artifacts) for areas with rich texture and sharp edges, especially when the bit-rate is relatively low. For some video codecs, there are built-in filters in the decoder to mitigate this problem. In HEVC, in-loop filtering is adopted, including a deblocking filter (DBF) and sample adaptive offset (SAO), to alleviate the block and ringing artifacts, respectively. However, the results are still not satisfactory when the bit-rate is low.

A number of approaches have been proposed to reduce these artifacts. Conventional approaches design filters based on image priors (Low-rank, non-local similarity, sparse). But most of these priors are hand-crafted and not optimal in some cases. With the success of convolutional neural networks (CNN) in image processing, CNN based algorithms have also been proposed. In [2], a compression artifacts reduction CNN (ARCNN) is proposed, which can achieve significant improvement compared to conventional approaches. In [3], a reconstruction network is proposed, which can solve both super-resolution and enhancement problems at the same time. In [7], a decoder-side HEVC quality enhancement using a scalable CNN is proposed, which can enhance the quality of Intra-frames and Inter-frames with different sub-networks.

However, most of these approaches are exploiting only spatial information, they are typically applied on the luminance component and the inter-channel correlation is not exploited. In image and video compression, the YUV420 color format is usually adopted, based on the assumption that the human visual system is not so sensitive to color differences compared to brightness changes. When the decoded RGB images are obtained, usually the G channel has the best quality, R and B have relatively low quality. In this condition we take the BPG codec and the Kodak dataset as an example to prove this statement. As shown in Fig. 1 and 2, for both Peak Signal-to-Noise Ratio (PSNR) and Multi-Scale Structural Similarity (MS-SSIM), the G channel shows higher quality even when the bit-rate is low. And for PSNR, when the bit-rate is higher, the gap is greater. For YUV444 format, we observed similar results, but the gap is smaller.

Based on these observations, and the inter-channel correlations, both structurally and spectrally, in this paper, we proposed a 3-stage CNN based approach to enhance the image quality. First, since the G channel has higher quality, a network is adopted to enhance the G channel using spatial information. Then, because of the strong inter-channel correlations, the enhanced G values are used to guide the enhancement of R and B separately in the second stage. Finally, the three channels are concatenated together and enhanced jointly in the third stage.

The major contributions of this work can be summarized as follows: First, the inter-channel correlation is exploited

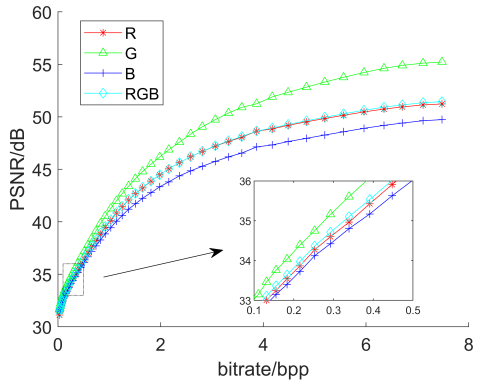


Figure 1. PSNR of R, G and B (BPG, YUV420, Kodak dataset)

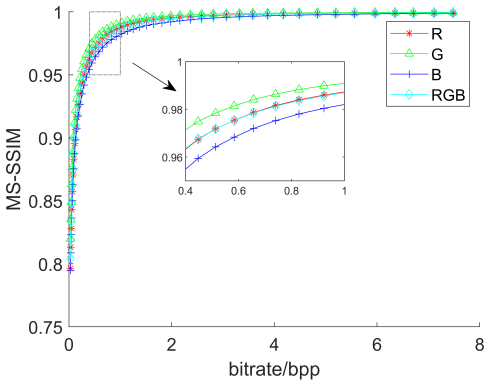


Figure 2. MS-SSIM of R, G and B (BPG, YUV420, Kodak dataset)

to enhance the quality of the decoded image. Second, a 3-stage structure is used, the image quality is enhanced stage by stage. Third, the proposed scheme is a post-processing approach and hence compatible with existing standard image and video codecs, which makes the approach applicable in practice.

2. Proposed scheme

In image and video compression, YUV420 is the most commonly used format. The U and V components are down-sampled both horizontally and vertically. This leads to the aforementioned characteristics of compressed images when transformed back to RGB domain. The R, G, and B channels exhibit strong inter-channel correlation, both structurally and spectrally, which means that the samples from other channels can be used to enhance the quality of the current channel. Based on these characteristics, we propose the 3-stage CNN structure shown in Fig. 3 for image quality enhancement.

First, the compressed image is decoded with a standard image codec. The first stage is designed to reconstruct the G channel. The $DecodedImage$ is split into $InputR$, $InputG$ and $InputB$, and the $InputG$ is fed into the first stage. Then, the output of the first stage $IntermediateG$ is concatenated with $InputR$ and $InputB$, respectively, and fed into the second stage. The second stage is designed to explore the correlations between R/G and G/B, using the high-quality $IntermediateG$ to guide the reconstruction of R and B. Using two separate networks in the second stage to reconstruct R/G and G/B is motivated by the differences in the inter-channel correlation of R/G and G/B. Two separate networks can better model and make the most of the inter-channel correlations. In the third stage, we concatenate the obtained intermediate R, G, B data, as the input of the third stage, where the inter-channel correlations are further exploited. Finally, the enhanced images are obtained from the third stage. The residual learning structure from [4] is used for each stage to boost the learning process.

Fig. 4 shows the detailed structure of the network unit for each stage. In the first layer, 128 filters of size $3 \times 3 \times d$ are used to generate feature maps, the last convolutional layer adopts d filters of size $3 \times 3 \times 128$ to generate the corresponding output. For the hidden layers, 128 filters of size $3 \times 3 \times 128$ are adopted. The number of the layers in each unit K is set to 5 and d is set to 1, 2, 3 in the three stages, respectively. Stride is set to 1, and zero-padding of size 1 is used to ensure that each feature map has the same size as the input.

Consider the training dataset $(\mathbf{X}_i, \mathbf{Y}_i)_{i=1}^N$, where $\mathbf{X}_i$ is the i-th decoded compressed image, $\mathbf{Y}_i$ is the corresponding ground-truth RGB image, and N is the number of images in the training data. During training, a loss function is defined to optimize the parameters of the networks. As shown in Fig. 3, four losses are defined for the proposed scheme. In the first stage, L_G is defined for the G channel. In the second stage, two loss functions L_{RG} and L_{GB} are defined since R/G and G/B are processed separately. In the third stage, L_{RGB} is defined as the loss for all three channels. The mean squared error (MSE) function is used as the loss function and the overall loss function used during training is defined as follows.

$$\begin{aligned} L(\omega_1, \omega_{21}, \omega_{22}, \omega_3) = \frac{1}{4}(L_G(\omega_1) + L_{RG}(\omega_1, \omega_{21}) + \\ L_{GB}(\omega_1, \omega_{22}) + L_{RGB}(\omega_1, \omega_{21}, \omega_{22}, \omega_3)) \\ L(\omega) = \frac{1}{N}\sum_{i=1}^{N}(\|\mathcal{F}(I_i;\omega) - O_i\|^2) \end{aligned} \quad (1)$$

where ω_j are the corresponding network parameters of the j-th stage. I_i and $\mathcal{F}(I_i;\omega)$ are the i-th input and output of each stage, and O_i is the corresponding ground-truth.

We also implemented a lite-version for complexity reasons. In this version, we simplify the model to 2-stage by removing the second stage. The first stage is for the G reconstruction, the RGB are reconstructed jointly in the second stage. We set the number of the feature maps to 64.

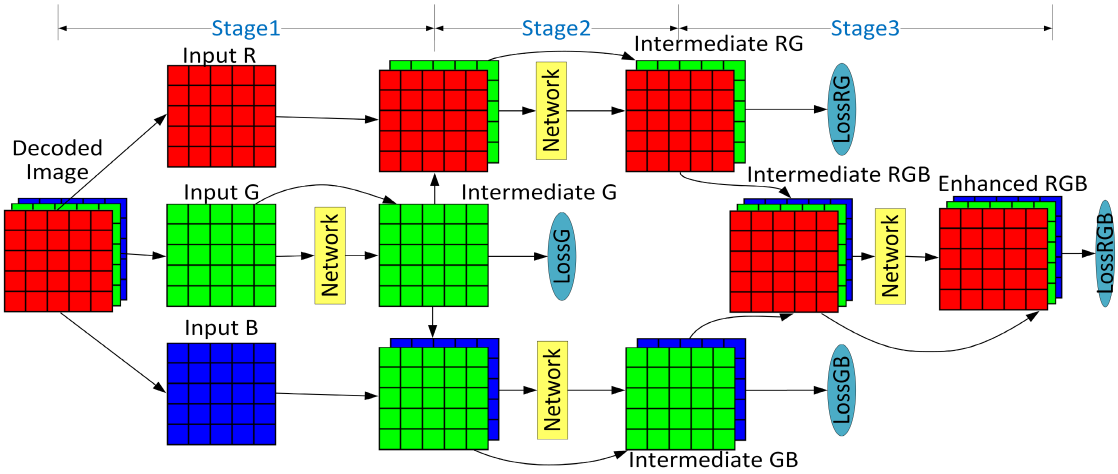


Figure 3. Structure of the proposed 3-stage CNN scheme

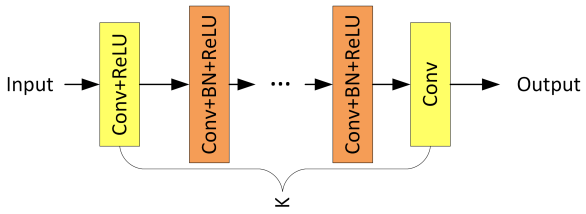


Figure 4. Structure of Network Unit

Other settings are the same as the 3-stage model. The lite-version is much faster than the 3-stage version, a detailed performance comparison will be presented in Section III.

3. Experiments and results

The provided training dataset is adopted in our experiments as training data. In this dataset, there are 1633 natural images of various scenes shot by mobile devices and professional cameras. We randomly pick 1000 images for the training dataset. The BPG codec is used to generate the compressed images. The quantization parameter (qp) is set to 38 due to the bit-rate constraint, jctvc option is enabled to achieve the best compression results, the level is set to 9. The patch size is set to 50×50, and the patches are non-overlapped. The mini-batch size is set to 64. The weights of the networks are initialized according to [4] and the Adam solver is used to optimize the parameters. The starting learning rate is 0.001, and divided by 5 every 5 epochs. There are 30 epochs in total. Other hyper-parameters are using the default settings from [5]. The training is performed using Matlab(2018a) with the Matconvnet [6] toolbox.

First, two example images from the validation dataset are shown in Fig. 5 to show the visual quality of the proposed method. Usually the texture-rich and sharp edge area are the challenging cases. We zoom in the tree part of the 'philipp-reiner-207' and the football player of the 'IMG_20161023_122645' from the given validation dataset. It can be seen that for BPG compression, block artifacts, false-color pixels and shadows can be observed along the edges of the trees and the football player. With the proposed method, for both the 2-stage and 3-stage version, these artifacts can be well eliminated and the visual quality is improved. The 3-stage approach leads to better image quality than the 2-stage.

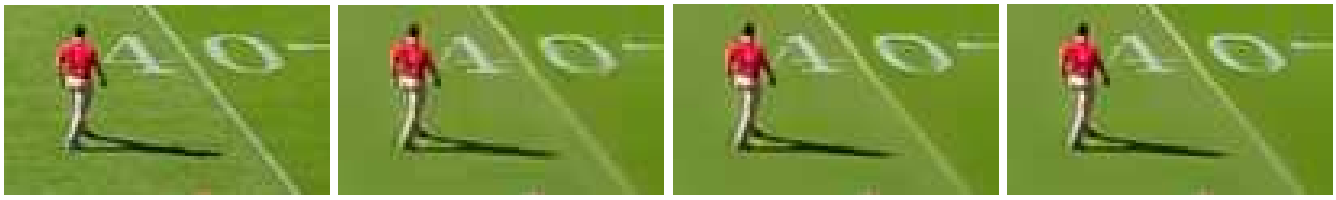
The average PSNR, composite PSNR (CPSNR) and MS-SSIM are adopted to evaluate the objective quality of the proposed approaches. A weighted PSNR is also adopted in this challenge, which computes a single Mean Squared Error (MSE) value by averaging across all RGB channels of all pixels of the whole dataset. From that value calculates a PSNR value, which is marked as the WPSNR in the table. The results are listed in Table 1. They are all under the constraint 0.15bpp required by the challenge.

Table 1. PSNR (in dB) and MS-SSIM results for the proposed 3-stage approach on the validation dataset

Evaluation	BPG	Ours-2stage	Ours-3stage
PSNR-R	31.22	31.85	31.94
PSNR-G	32.25	32.75	32.81
PSNR-B	30.98	31.78	31.90
CPSNR	31.43	32.08	32.17
WPSNR	30.85	31.48	31.57
MS-SSIM	0.948	0.954	0.955

From the results, it can be seen that the proposed 2-stage and 3-stage methods lead to 0.6-0.7dB PSNR and 0.006-0.007 MS-SSIM improvements on the validation datasets in comparison to the BPG baseline with default settings. In terms of the PSNR of each color component, the G channel has about 0.5dB improvement, which proves that the first stage works, the improvements of R and B are 0.7dB and 0.9dB, higher than that of G, which proves the rationality

(a) Ground Truth (PSNR / SSIM) (b) BPG (33.21dB / 0.9140) (c) 2-stage (33.63dB / 0.9346) (d) 3-stage (33.68dB / 0.9375)



(e) Ground Truth (PSNR / SSIM) (f) BPG (33.46dB / 0.9584) (g) 2-stage (33.68dB / 0.9635) (h) 3-stage (33.83dB / 0.9648)

Figure 5. Visual Quality Comparison (Best seen on a computer monitor)

and effectiveness of the proposed network.

4. CLIC2018 Image Compression Challenge

This approach is proposed for the CLIC2018 challenge. We submitted two versions of our decoder to the evaluation server, the 2-stage version and the 3-stage version. The 2-stage version is about five times faster than the 3-stage version, but the 3-stage version achieves the better quality. The BPG decoder is implemented by python binding with the shared objects libbpg.so compiled from BPG source code. The enhancement network is implemented with Tensorflow and the saved network parameters. Because the evaluation server provides only CPU, the Tensorflow is running in CPU mode. We also notice that the running time on the evaluation server is much longer than that on the local machine. For the 2-stage version, we need about 1500 seconds (i7-4770 CPU and 8GB RAM) for all 102 images locally, but it took more than 2 hours (2 CPUs, 8GB RAM and 8GB Swap) on the server. For the 3-stage version, it is the same situation. This may be caused by high load of the server or the resource assignment strategy of Tensorflow when the hardware configuration is different. For encoding, we use the standard BPG encoder compiled from the source code to encode all the images to the compressed format. Because there's an overall bit-rate constraint, 0.15bpp, for simplicity, we set the qp to 38 for all the images.

5. Conclusion

This paper presents a 3-stage CNN-based decoder side image enhancement scheme. The first stage is used to enhance the G channel by using the spatial information. The second and the third stage are used to exploit the inter-channel correlation. They use the enhanced G to guide the enhancement of R and B. The experimental results on both validation and test datasets show that the proposed scheme leads to noticeable improvements compared to the BPG baseline. Also, the proposed method is built on top of the standard image codec, which makes that it is compatible with any existing image codec. As a part of the future work, we will investigate the potential gains applying the proposed scheme to video codecs.

References

[1] F. Bellard. The bpg image format. http://bellard.org/bpg/. 1

[2] C. Dong, Y. Deng, C. Change Loy, and X. Tang. Compression artifacts reduction by a deep convolutional network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 576–584, 2015. 1

[3] F. Jiang, W. Tao, S. Liu, J. Ren, X. Guo, and D. Zhao. An end-to-end compression framework based on convolutional neural networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017. 1

[4] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1646–1654, June 2016. 2, 3

[5] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, May 2015. 3

[6] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692. ACM, 2015. 3

[7] R. Yang, M. Xu, and Z. Wang. Decoder-side hevc quality enhancement with scalable convolutional neural network. In *Multimedia and Expo (ICME), 2017 IEEE International Conference on*, pages 817–822. IEEE, 2017. 1