Learned Image Compression with Residual Coding

Wei-Cheng Lee[1] David Alexandre[1] Chih-Peng Chang[2] Wen-Hsiao Peng[2]
Cheng-Yen Yang[1] Hsueh-Ming Hang[1]

[1]Department of Electronics Engineering
[2]Department of Computer Science and Information Engineering
National Chiao Tung University, Taiwan

a1357960603@gmail.com, {davidalexandre.eed05g, gcwhiteshadow.cs04}@nctu.edu.tw
wpeng@cs.nctu.edu.tw, chenyen.ee99g@g2.nctu.edu.tw, hmhang@nctu.edu.tw

Abstract

We propose a two-layer image compression system consisting of a base-layer BPG codec and a learning-based residual layer codec. This proposal is submitted to the Challenge on Learned Image Compression (CLIC) in April 2019. Our contribution is to integrate several known components together to produce a result better than the original individual components. Also, unlike the conventional two-layer coding, our encoder and decoder take inputs also from the base-layer decoder. In addition, we create a refinement network to integrate the residual-layer decoded residual image and the base-layer decoded image together to form the final reconstructed image. Our simulation results indicate that the transmitted feature maps are fairly uncorrelated to the original image because the object boundary information can be provided by base-layer image. The experiments show that the proposed system achieves better performance than BPG subjectively at the given 0.15 bit-per-pixel constraint.

1. Introduction

Recent researches reveal that the deep learning base image compression methods can potentially produce better quality images than the traditional coding scheme.

In participating in the low-rate track of CLIC 2019, we propose a hybrid coding scheme, which consists of a BPG codec as the base-layer, and a residual-layer, which uses an autoencoder architecture to reduce the artifacts generated by the BPG codec at lower bitrates. This method is inspired by the conventional layered coding concept and the recent deep-learning based codec in [12] and [3]. At the receiver, the decoder network combines the intermediate outputs of base-layer and residual-layer, and then a refinement network resynthesizes the output image. This refinement network can be considered as a post-processing component to further improve the reconstructed image quality. More specifically, our system allocates about 0.09 bit-per-pixel (bpp) to the BPG coding and about 0.06 bpp to the residual coding to meet the 0.15 bpp target.

Although the multi-layer coding concept exists in the conventional coding system, our contribution is to integrate several known components together to produce a result better than the original individual components. Also, unlike the conventional two-layer coding, our encoder and decoder take inputs also from the base-layer decoder. In addition, we create a refinement network to integrate the residual-layer decoded *residual* image and the base-layer decoded image together to form the final reconstructed image. Our simulation results indicate that the transmitted feature maps are fairly uncorrelated to the original image because the object boundary information can be provided by base-layer image.

2. Related work

The recent learning-based image compression methods [4, 5, 6, 9, 12, 13, 14, 15] use an autoencoder to extract the latent representation of input data. And it is trained to strike a balance between distortion and rate losses in an end-to-end manner. Another common framework is a hybrid CNN-assisted structure [3] which typically consists of a base-layer processing by the traditional compression algorithm and a learning-based residual layer. In addition to the post-processing, multi-layer coding [7, 10, 11] is widely adopted in compression research for the artifact reduction.

A high efficient entropy coder and an accurate rate estimator have played a key role in the compression for decades, and most of the learning-based compression methods adopt such techniques in their system. The binary arithmetic codec with a context model is adopted by [4, 5]. The model in [4] is a hand-crafted design and [9, 14] use the

learned context models.

3. Proposed Methods

3.1. Overall architecture and design philosophy

Fig.1 shows the proposed residual image compression system. There are two data paths, the base-layer is a conventional image compression method, such as BPG, and the residual-layer is a learning based scheme to encode the residual errors. In fact, this component is more complicated than the traditional residual layer because it has three component: residue *encoder*, residue *decoder*, and the refinement layer. One can observe that the residue encoder takes in the residual signal and also it takes in the reconstructed base-layer (BPG) images. The idea is that the decoded base-layer image is also available at the decoder. Therefore, it may be able to provide some "side-information" of the residual image and thus the transmitted latent variables (feature maps) do not need to contain this "side-imformation".

The Decoder is kind of the reverse of the decoder; however, there is an additional Refinement Network, which has two inputs: the base-layer recovered image and the output of residue decoder. Therefore, the Refinement Network plays the role of postprocessing. However, because it is included in the training loop, its function is more than summing up the base-layer image and the enhancement residuals.

Fig.2 illustrates the various steps in this processing flow. Fig.2-(a) is the original input image $\boldsymbol{x}$; Fig.2-(b) is the base-layer (BPG) decoded image $\boldsymbol{x_c}$; Fig.2-(c) is the residual (difference) image at the encoder ($\boldsymbol{x} - \boldsymbol{x_c}$), and the residual image zero value is shifted to 128 for display; Fig.2-(d) is the feature maps before quantization; Fig.2-(e) is the quantized feature maps (two bitplanes each), which are transmitted to the receiver. It reveals that the future maps are pretty much uncorrelated to the original image. It shows that our autoencoder indeed tends to compress the most meaningful information instead of the location information offered by BPG decoded image. Fig.2-(f) is the decoded residual image; and finally, Fig.2-(g) is the Refinement Network output, our reconstructed image.

Since the residual image has typically the properties of small variation and sparsity, we hope the residual coding and the refinement network can enhance the reconstructed image quality at low bitrates. With the aids of BPG decoded image as the additional information, the autoencoder can predict where the residual might have high response more precisely so that it should only compress the most meaningful information. In our structure, the entire encoder produce the BPG codes and residual codes. Our learning based residue encdoer and decoder originate from the autoencoder in [12] with modification.

There are two challenge issues in designing the autoencder here. (1) The quantization process on the feature maps is non-differentiable. (2) The system aims to optimize both the compression rate and the distortion, where the entropy rate defined on discrete codes is also a non-differentiable term.

To overcome the above problems, we adopt the soft bit conversion, an approximation to quantization mapping, and a hand-crafted context model in the training phase. Both designs come from Alexandre *et al*. [4]. With the introduction of soft bit conversion and rate estimator, all the networks can be jointly optimized in an end-to-end manner.

3.2. Detailed structure and loss function

The input to the encoder is a six-channel tensor, which a residual image concatenates a BPG decoded image in RGB format. Given an input image $\boldsymbol{x}$, the BPG encoder produces a compressed image $\boldsymbol{x_c}$. With both $\boldsymbol{x_c}$ and its corresponding residual image $\boldsymbol{x} - \boldsymbol{x_c}$, a learning based encoder generates a set of residual feature maps $z = E(x, x_c)$, which the value are normalized to the range of 0 to 1. The normalization at the beginning of the encoder restricts the tensor input values to $[-1, 1]$. In order to obtain $\hat{\boldsymbol{z}}$ from $q = Q(z)$ by the quantizer Q and its inverse operation $\hat{z} = IQ(q)$, we adopt a simple scalar quantization to reduce the number of bits for representing the residual feature maps z. It keeps the first d bits after the decimal point in testing, and it keeps the first d bits after soft bit conversion in training, which is a differentiable approximation to the scalar quantization, respectively. Then, we arrange q into bit planes. We adopt the context-adaptive bit-plane encoder/decoder (CABIC/CABID) proposed in [4] to encode the bit planes. At the beginning of decoding stage, the residual feature sample is first recovered by inverse quantization, and then the refinement subnets generates the final reconstructed image $\hat{x} = R(D(\hat{z}, x_c), x_c)$ using the inputs of both decoded BPG image and decoded residuals.

The data flow with dashed arrow lines shown in Fig. 1. The soft bit conversion enables a differentiable distortion objective loss function in the training phase. In this work, the loss function is defined as:

$$\mathcal{L} = \lambda \times L_R(\boldsymbol{q}) + L_D(\boldsymbol{x}, \hat{\boldsymbol{x}}), \tag{1}$$

where $L_D(\boldsymbol{x}, \hat{\boldsymbol{x}})$ is define by some measure of reconstruction error, such as mean square error (MSE) or MS-SSIM, and $L_R(\boldsymbol{q})$ is rate loss predicted by CNN-based rate estimator.

As for the rate estimator, the actual bitrate is roughly proportional to the entropy of the quantized feature maps. To estimate the entropy value, we adopt the rate estimation model in [4] and factorize the distribution $p(\hat{\boldsymbol{q}})$ as a product of conditional distributions

$$p(\boldsymbol{q}) = \prod_{i=1}^{n} p(q_i | q_{i-1}, ...q_1), \tag{2}$$

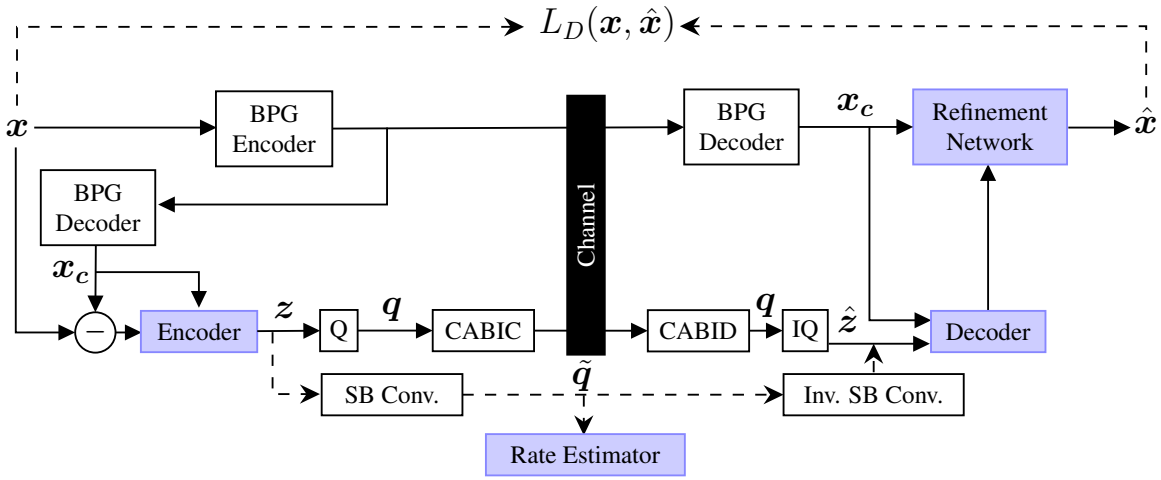

Figure 1. The architecture of the proposed image compression system with residual coding.

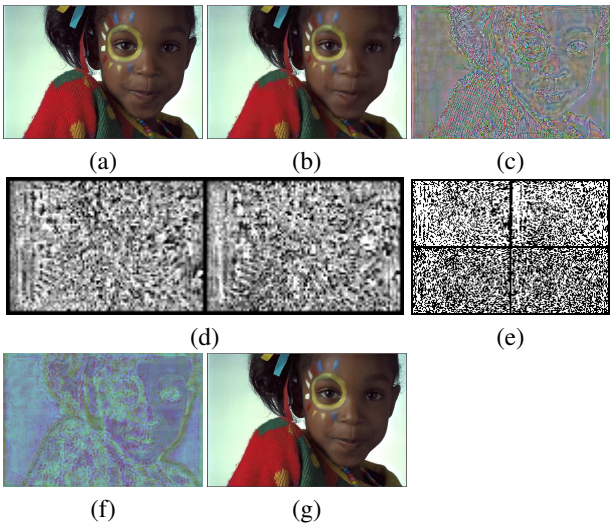

Figure 2. Visualization of each step in our sysem. BPG based-layer: 0.084 bpp, MS-SSIM: 0.926, PSNR: 29.84dB; Residual-layer: 0.1464 bpp, MS-SSIM: 0.948, PSNR: 30.51dB

where the samples of quantized feature maps $\boldsymbol{q}$ are indexed in the raster scan order.

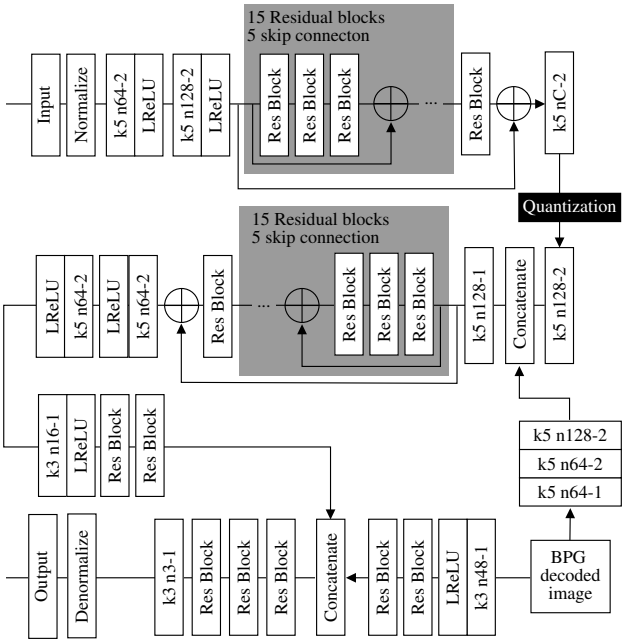

Figure 3. The architecture of proposed autoencoder and refinement network

4. Experiments

4.1. Autoencoder and refinement network

The proposed autoencoder and refinement network is shown in Fig 3, in which the upper and middle parts are the encoder E and decoder D respectively. And the lower part is the refinement subnet R. The E and D we used has similar architecture in [12]. The R can be regarded as a feature extraction network with 16 channels and 48 channels for residuals and BPG decoded image respectively, and then it resynthesizes a refined output image. The difference is that we adopt two feature maps to achieve the constraint of 0.15 bpp on the average for the entire CLIC test set. For the convolution layers in E, R, the notation "k5 n64-2" represents a convolution layer with kernel size 5, 64 output channels, and a stride of 2, and the corresponding deconvolution layers constitute D. Note that all the convolution layers are using SAME padding and followed by a batch normalization layer and activation function. The $Concatenate$ layer stacks up two input tensors instead of summation. The input of E is normalized by the $Normalize$ layer to zero mean and unit variance with respect to the statistics of training set, and the $Denormalize$ layer reverts the normalization process before the output layer of R.

Model	Bits/Pixel	PSNR	MS-SSIM
BPG	0.1497	28.391	0.939
Ours(MSE)	0.1483	27.367	0.934
Ours(MS-SSIM)	0.1491	27.485	0.940

Table 1. Compression result comparison on CLIC test dataset.

Model	Bits/Pixel	PSNR	MS-SSIM
BPG	0.1508	27.443	0.922
Ours	0.1493	26.882	0.927
Ours w/o R	0.1502	26.668	0.922

Table 2. Comparison the importance of refinement network on Kodak dataset.

4.2. Training

Our training dataset contains 1672 images provided by CLIC 2019 [1]. In the training phase, we randomly crop these images into 256×256 patches as inputs of our autoencoder, and we use the Kodak dataset [2] and the CLIC 2019 test set for validation and testing respectively.

The training procedure for autoencoder, refinement nets, and the rate estimator are divided into two phases. The first phase is to calculate the statistics of context probabilities (ctx for short) from the feature maps z, and then optimize the CNN-based rate estimator by minimizing L2 loss between $p(q_i|ctx)$ and $p(\tilde{q}_i|ctx)$. In the second phase, we update the parameters in the autoencoder and the refinement nets using (1).

We use Adam optimizer [8] with a mini-batch size 8 to train two models using different measure of distortion, MSE and MS-SSIM. Setting a learning rate at $1 \cdot 10^{-5}$ for the initial value and we decay it by a factor of 10 for every 200 epochs. In order to prevent the rate distortion value in (1) changes dramatically during the training process, we turn off $L_R(\boldsymbol{q})$ term by setting λ to 0 in the first 200 epoch.

4.3. Experiment results

Table 1 shows the comparison of the aforementioned methods using MSE and MS-SSIM as the distortion metrics individually at 0.15 bpp on the test set released by CLIC 2019. In our system, using MS-SSIM as the distortion objective gives a better performance than using MSE both objectively and subjectively. Although the BPG has around 1dB higher than ours in terms of PSNR, our model slightly outperforms on the MS-SSIM compared to the BPG. Fig.4 compares the compressed images at about the same rate using BPG alone and using out system. To see their differences more clearly, we also show the error images between the original and the coded ones in Fig.4(b) and (d). To visualize the error images, they are scaled by a factor of 5 and the zero value is shifted to 128. These two error images have somewhat different characteristics.

Table 2 shows the effectiveness of the refinement network. In this comparison, our model is trained for MS-SSIM with and without the refinement network. In the case of no refinement net, we replace the refinement network by the addition operator on the BPG decoded image and the decoded residuals. With the refinement network, both PSNR and MS-SSIM are improved.

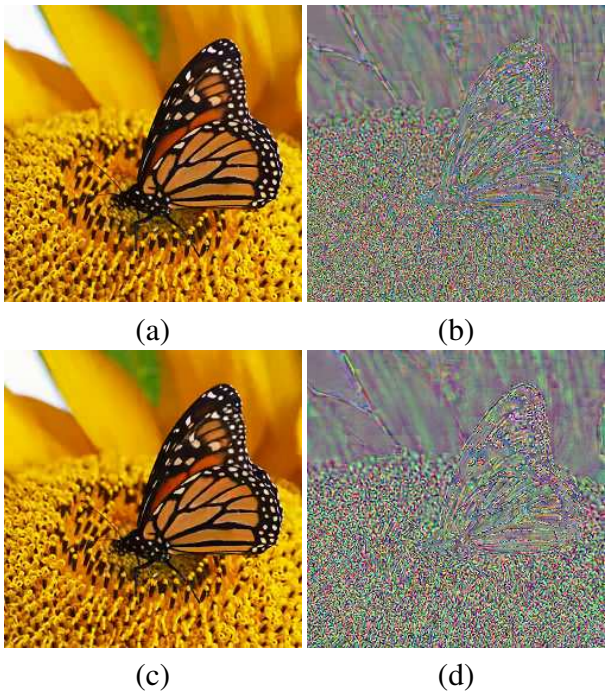

Figure 4. Comparison between images using BPG and using our system at about same rate. (a)BPG: 0.144 bpp, MS-SSIM 0.915, PSNR 27.661dB; (b)coding errors of (a); (c)Our system: 0.152 bpp, MS-SSIM 0.934, PSNR 26.779dB; (d)coding errors of (c).

5. Conclusions

In this paper, we propose an end-to-end trainable hybrid image coding scheme. The base-layer is the standard BPG codec, while the residual layer is a deep-learning based codec. In addition, we modify the inputs of both learning-based encoder and decoder, and we insert a refinement net to synthesize the final reconstructed image. This hybrid scheme has the flexibility of bitrate control; that is, we can adjust the bitrate split between the base-layer and the residual-layer to meet the target bitrate. Our method slightly outperforms BPG on MS-SSIM, it still has a somewhat lower PSNR. We believe there is a good potential of this hybrid structure for image compression, and may be extended to video compression.

6. Acknowledgment

This work was supported in part by Ministry of Science and Technology, Taiwan under Grant MOST 108-2634-F-009-007 through Pervasive AL Research (PAIR) Labs, Na-

tional Chiao Tung University, Taiwan.

References

[1] Challenge on Learned Image Compression. http://compression.cc/. 4

[2] Kodak PhotoCD dataset. http://r0k.us/graphics/kodak/. 4

[3] Mohammad Akbari, Jie Liang, and Jingning Han. Dsslic: deep semantic segmentation-based layered image compression. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2042–2046. IEEE, 2019. 1

[4] D. Alexandre, C.-P. Chang, W.-H. Peng, and H.-M. Hang. Learned image compression with soft bit-based rate-distortion optimization. *arXiv preprint arXiv:1905.00190*, 2019. 1, 2

[5] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016. 1

[6] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018. 1

[7] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. Compression artifacts reduction by a deep convolutional network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 576–584, 2015. 1

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[9] Mu Li, Wangmeng Zuo, Shuhang Gu, Debin Zhao, and David Zhang. Learning convolutional networks for content-weighted image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3214–3223, 2018. 1

[10] Yue Li, Dong Liu, Houqiang Li, Li Li, Feng Wu, Hong Zhang, and Haitao Yang. Convolutional neural network-based block up-sampling for intra frame coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(9):2316–2330, 2018. 1

[11] Zhi Liu and Chunhong Cui. A new low bit-rate coding scheme for ultra high definition video based on super-resolution reconstruction. In *2018 IEEE International Conference on Computer and Communication Engineering Technology (CCET)*, pages 325–329. IEEE, 2018. 1

[12] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool. Conditional probability models for deep image compression. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 3, 2018. 1, 2, 3

[13] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10771–10780, 2018. 1

[14] Oren Rippel and Lubomir Bourdev. Real-time adaptive image compression. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2922–2930. JMLR. org, 2017. 1

[15] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017. 1