Ultra Low Bitrate Learned Image Compression by Selective Detail Decoding

Hiroaki Akutsu Akifumi Suzuki
Hitachi, Ltd., Japan
hiroaki.akutsu.cs@hitachi.com

Zhisheng Zhong Kiyoharu Aizawa
The University of Tokyo, Japan

Abstract

Neural network-based learned image compression has a special feature in that a differentiable image quality index can be used as a loss function directly, and a decoder and an encoder can be optimized by the quality index through end-to-end learning. From a perceptual view, we hypothesized that there were detailed important parts in pictures. For those parts, we applied an additional decoder and weighted loss function to achieve both low bitrate image compression and perceptual quality. Furthermore, our approach can automatically determine which region an additional decoder will take for an input image. Experiments visually showed that the proposed method can recognize important parts, such as text and faces, and we show that our method can decode images more clearly than the simple MS-SSIM training model.

1. Introduction

Image compression configured by neural networks in an end-to-end manner has been studied [9] [10] [11]. In general, neural network-based learned image compression is composed of an encoder, a decoder, a quantizer, an entropy estimator, and an adaptive arithmetic coder, the same as traditional image compression. In many cases, the encoder and the decoder are composed of a convolutional auto-encoder, and the entropy estimator is also composed of a neural network. One of the advantages of neural network-based learned image compression is that a differentiable image quality index can be directly used as a loss function of the neural network. In past research, PSNR (MSE), MS-SSIM [13], etc. are often used as image quality indexes. Another advantage is that a network structure researched and optimized by other tasks (recognition, segmentation, generation model, super-resolution, etc.) could be commonly used as a compression component. For example, in adaptive arithmetic coding, a high compression rate is achieved by dynamically predicting the probability distribution of a quantized value. There are papers [10] [11] that use neural networks such as auto-regression models, e.g., PixelCNN [12], and auto-encoders to generate hyperpriors for this prediction. There is one study [1] that uses GANs for image compression. By using GANs, it is possible to output a likely image even at a low bitrate, which is difficult with conventional image compression. However, as stated in this paper, there is a problem that symbolic information would collapse.

We aim to realize data compression technology that maintains perceptual quality while keeping the bitrate as low as possible. Our contribution is three points: (i) we show the MS-SSIM for entire image and perceptual quality do not always match, and we propose to apply two decoders to achieve both low bitrate and high perceptual quality that a main decoder is trained by conditional GAN-style and a selective detail decoder is trained by weighted MS-SSIM loss [3], it should be notable that we modify the decoder part without changing the encoder, (ii) the adding of causal attention modules to the context entropy estimator for further reducing bpps with respect to one of the state-of-the-art auto-regressive models [5], and (iii) super-resolution-style encoder and decoder blocks [14] with spatial attention modules.

2. Proposal Method

2.1. Overview

Figure 1 shows the overall network configuration of the proposed method. The method consists of an encoder E, a main decoder G_m, a selective detail decoder G_s, quantizers Q, an entropy estimator H, a discriminator D, adaptive arithmetic encoders (AE), and decoders (AD). E, G_m, G_s, H, and D have parameters to train, and Q, AE, and AD have no parameters. We used a round quantizer [4] and adaptive range coders as AE and AD. Assume an input image is $\mathbf{x}$, and a quantized feature map $\hat{\mathbf{z}}$ to be arithmetically encoded is obtained with $\hat{\mathbf{z}} = Q(E(\mathbf{x}))$. The entropy estimator H takes as input the unquantized feature maps $\mathbf{z}$ and $\hat{\mathbf{z}}$, and it outputs the parameters of the probability distribution required for adaptive arithmetic coding.

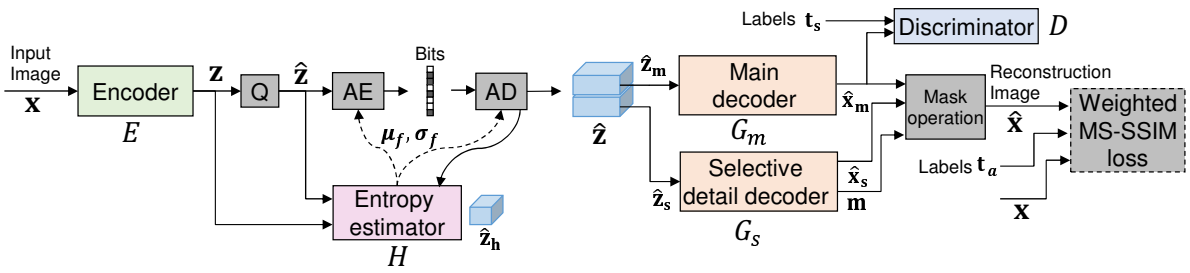

Figure 1. Network architecture overview.

2.2. Main Decoder and Selective Detail Decoder

The proposed method uses two decoder, G_m and G_s. The main decoder G_m is a model for decoding entire images $\hat{\mathbf{x}}_\mathbf{m}$, and it is trained adversarially with the discriminator network D. The selective detail decoder G_s is a model specialized for specific important parts and trained using weighted MS-SSIM [3]. We used two type of labels $\mathbf{t_a}$ and $\mathbf{t_s}$ for training which have same spatial dimensions (width and height) with images in the training dataset. The weighted MS-SSIM uses annotation information labels $\mathbf{t_a}$ that indicates the important parts to be processed by G_s. We chose scene texts and human faces as the important parts, which are labeled by $\mathbf{t_a}$. D is used for the training of G_m and not used for G_s. D uses semantic segmentation labels $\mathbf{t_s}$ with images $\hat{\mathbf{x}}_\mathbf{m}$ as inputs like conditional GANs. In conventional conditional GAN-based image compression [1], the label is also input to the generator side, but we found it works even if it is removed. Thus, our compression network architecture have two decoders with keeping a single encoder. This approach is general and does not require additional labels when compressing and decompressing data.

The mask vector $\mathbf{m}$ is the output of one channel of the G_s with the sigmoid function applied, and it indicates which part of the output image $\hat{\mathbf{x}}_\mathbf{s}$ from G_s should be used. It reconstructs the image $\hat{\mathbf{x}}$ by following the mask operation,

$$\hat{\mathbf{x}} = \hat{\mathbf{x}}_\mathbf{s} \odot \mathbf{m} + \hat{\mathbf{x}}_\mathbf{m} \odot (1 - \mathbf{m}). \tag{1}$$

With end-to-end learning, we aim to improve the total perceptual quality by automatically selecting which part of a picture is processed by which decoder with $\mathbf{m}$.

2.3. Entropy Estimator with Causal Attention

The entropy estimator of the proposed method uses both context and hyperprior-based prediction, and it finally outputs the parameters of the probability distribution of $\hat{\mathbf{z}}$ values, as in the same approach of [11]. In particular, to improve the prediction of the probability distribution of the values, we propose applying causal attention modules to the context estimator and the mixing body that is the part that mixes hyperpriors and the context estimation.

Figure 2 shows the structure of our entropy estimator. The entropy estimator inputs $\mathbf{z}$ and $\hat{\mathbf{z}}$ and outputs the parameters $\boldsymbol{\mu_f}$, $\boldsymbol{\sigma_f}$ of Gaussian distributions that represent distributions of $\hat{\mathbf{z}}$ values. The hyperprior encoder H_e encodes $\mathbf{z}$ and then quantizes them to get $\hat{\mathbf{z}}_\mathbf{h}$. $\hat{\mathbf{z}}_\mathbf{h}$ is encoded into a very small bit stream by adaptive arithmetic coding and stored as compressed data along with $\hat{\mathbf{z}}$. Using $\boldsymbol{\mu_f}$, $\boldsymbol{\sigma_f}$, the entropy is estimated by the following that uses the cumulative distribution function of the Gaussian distribution,

$$I_f(\hat{\mathbf{z}}) = -\sum \log\Big(\frac{1}{2}\mathrm{erf}\Big(\frac{\hat{\mathbf{z}} - \boldsymbol{\mu_f} + 0.5}{\sqrt{2}\boldsymbol{\sigma_f}}\Big) - \frac{1}{2}\mathrm{erf}\Big(\frac{\hat{\mathbf{z}} - \boldsymbol{\mu_f} - 0.5}{\sqrt{2}\boldsymbol{\sigma_f}}\Big)\Big). \tag{2}$$

The hyperprior entropy I_h can be obtained in the same way as I_f. The final entropy is calculated by $I(\hat{\mathbf{z}}, \mathbf{z}) = I_f(\hat{\mathbf{z}}) + I_h(Q(H_e(\mathbf{z})))$. Figure 2 also shows the structures of the context estimator and the mixing body. There is one proposal [10] of using residual blocks for the context estimator. For further improvement, we refer to PixelSNAIL [5], and we apply causal attention modules to the context estimator and the mixing body.

2.4. Super-Resolution Style Autoencoder Modules

In general, image compression with the convolutional auto-encoder uses a large convolution kernel with a stride of 2 to reduce the spatial dimensions by $1/2$ repeatedly [10] [11]. We proposed [14] using a technique called PixelShuffling [8] with residual channel attention blocks for the image compression area, which is often used for super-resolution tasks, without using a large kernel-size convolution.

Figure 3 shows the configuration of the building blocks used in this paper. In this paper, we use blocks obtained by adding attention in the spatial direction to the proposed method like RAMs [7]. The encoder body is a component that reduces the spatial dimensions by $1/2$, and it is used for E, H_e, and D. The decoder body is a component that increases the spatial dimensions by 2 times, and it is used for G_s, G_m, and H_d.

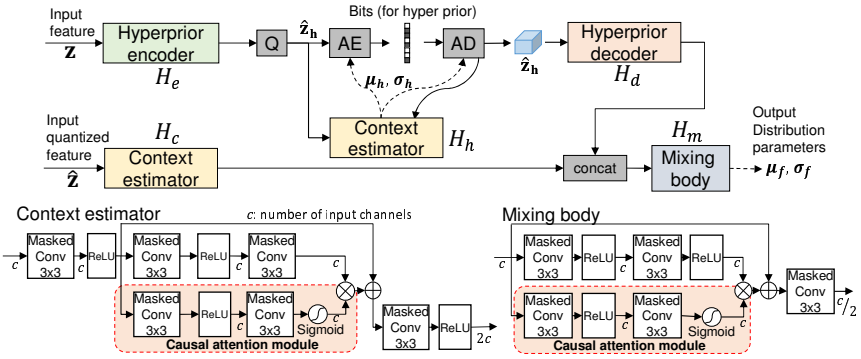

Figure 2. Entropy estimator.

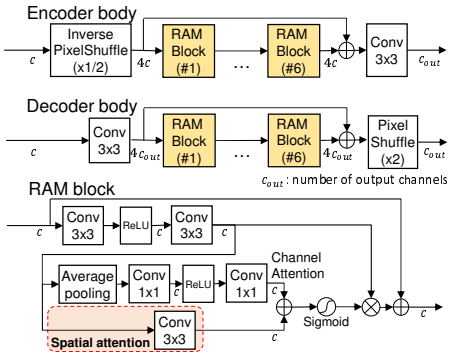

Figure 3. Network building blocks.

2.5. Loss Functions

In [1], a method was proposed of learning by combining an image distortion function, discriminator loss, and entropy loss with Lagrange multipliers. The differences with our proposed method are the following two. First, to automatically process each region of an image with different decoders, the weighted MS-SSIM loss function [3] is used for training. Second, semantic segmentation labels $\mathbf{t_s}$ are used only for the input of the discriminator and not for the encoder or decoder input. Our approach requires the labels $\mathbf{t_s}, \mathbf{t_a}$ during model training but not during compression and decompression after model training. This is because the model jointly learns these features by using the loss function and the discriminator.

Our entropy loss is represented by the $\mathcal{L}_e = \mathbb{E}\left[I(\hat{\mathbf{z}}, \mathbf{z})\right]$, and the distortion loss is given by

$$\begin{aligned}\mathcal{L}_d =& \mathbb{E}\left[1 - MSSSIM(\mathbf{x}, \hat{\mathbf{x}}_\mathbf{m})\right] + \\ & \lambda_p \mathbb{E}\left[1 - wMSSSIM(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{t_a})\right].\end{aligned} \tag{3}$$

For the main decoder output images $\hat{\mathbf{x}}_\mathbf{m}$, not only the discriminator but also the normal MS-SSIM is backpropagated to remove some visual discomfort. However, the hyperparameter λ_p is set to give priority to backpropagate the weighted MS-SSIM to the specific important region given by the annotation information $\mathbf{t_a}$.

In our configurations, stable training was possible by using a softplus-based discriminator loss function. We define r with $r = -1$ at discriminator phase and $r = 1$ at generator phase, the discriminator loss we used is given by

$$\mathcal{L}_g = \mathbb{E}\left[\log(1 + e^{D(\mathbf{x}, \mathbf{t_s})})\right] + \mathbb{E}\left[\log(1 + e^{rD(\hat{\mathbf{x}}_\mathbf{m}, \mathbf{t_s})})\right]. \tag{4}$$

Finally, the total loss function is given by

$$\min_{\theta_{G_{t,s},E,H}} \min_{\theta_D} V(\theta_{G_{t,s},E,H}, \theta_D) = \mathcal{L}_d + \lambda_e \mathcal{L}_e + \lambda_g \mathcal{L}_g. \tag{5}$$

3. Experimental Results

3.1. Experimental Conditions

The encoder E was composed of four encoder bodies, and a 3x3 conv layer was added at the end. The number of channels between these components were respectively 3, 32, 64, 128, 192, and 64. G_m was composed of 4 encoder bodies, and a 3x3 conv layer was inserted first. The numbers of channels among these components were respectively 32, 192, 128, 64, 32, and 3. G_s was similarly configured, but the numbers of channels among components were respectively 32, 192, 96, 64, 32, and 4. H_e was composed of two encoder bodies, and a 3x3 conv layer was inserted first. The numbers of channels among these components were respectively 64, 32, 32, and 32. H_d was composed of two decoder bodies, and a 3x3 conv layer was inserted last. The numbers of channels among these components were respectively 32, 32, 32, and 128. The configuration of the other components of the entropy estimator is as described in Figure 2.

The discriminator D was composed of four encoder bodies, and a 3x3 conv layer was added at the end. The numbers of channels among these components were respectively 8, 32, 64, 128, 192, and 1. The input of the images was 3 channels, and the remaining 5 channels were used for the input of label $\mathbf{t_s}$ in D. The input labels were one-hot expressions, and additional 1x1 convolution networks with a final output of 5 channels were added to reduce the label dimension.

We used images from the Open Images Challenge 2018 dataset [2] for training. For those images, semantic segmentation for $\mathbf{t_s}$ was machine generated, and annotations of faces and text parts for $\mathbf{t_a}$ were also machine generated, and those were used for training. The hyperparameters used were $\lambda_p = 5$, $\lambda_e = 0.18$, and $\lambda_g = 0.01$. We used the ADAM optimizer and set the learning rate as 8e-5. The batch size was 8 and the number of training iterations was about 800,000.

bpp: 0.219
MS-SSIM: 0.951 (0.959)

bpp: 0.216
MS-SSIM: 0.937 (0.950)

(a) ground truth

bpp: 0.125
MS-SSIM: 0.977 (0.957)
(b) MS-SSIM training

bpp: 0.117
MS-SSIM: 0.970 (0.942)
(c) proposal ($\hat{\mathbf{x}}$)

(d) The main decoder output ($\hat{\mathbf{x}}_\mathbf{m}$)

(e) The selective detail decoder output ($\hat{\mathbf{x}}_\mathbf{s} \odot \mathbf{m}$)

Figure 4. Experimental results using kodim18 and 20 (the pictures were clipped, and values in parentheses are clipping parts MS-SSIM).

bpp	MS-SSIM	PSNR
0.148	0.9648	28.028

Table 1. Evaluation results using the CLIC2020 validation dataset.

3.2. Results

The evaluation results using the Kodak PhotoCD dataset [6] are shown in Figure 4. (b) shows the results of evaluation using a model which was simply trained by the MS-SSIM loss function. The network configuration was the same as that of the proposed method, but the only decoder that was used was G_m. Under the same level bitrate condition, the proposed method decoded parts such as ground grass and textures of background objects more clearly compared with the simple MS-SSIM model. In addition, the face and characters are collapsed in image (d) $\hat{\mathbf{x}}_\mathbf{m}$ trained by the discriminator, but according to (e), the proposed method successfully masked the relevant part as a important part. It can be seen that it generated an image visually closer to the ground truth of (a). Furthermore, as an effect of the weighted MS-SSIM, the important parts were sharper than with the simple MS-SSIM model. Furthermore, the masked boundaries are not visually noticeable due to the effect of multi-scale filtering of the weighted MS-SSIM. As shown in the results, MS-SSIM for the entire image and the perceptual quality do not always match.

Finally, Table 1 shows the results using the CLIC2020 dataset. Our submission team's name is "neuro".

4. Conclusion

GAN-based image compression can achieve an ultra low bitrate, but in some cases, it can be visually inferior. To achieve visually superior, the proposed methods using two decoders, the main decoder trained on the GAN-base and the selective detail decoder trained on weighted MS-SSIM for areas specified by annotation information. Our method do not need those additional labels during compression and decompression processing after the model training. Furthermore, we leverage a network capable of outputting high-definition images used in the super-resolution area, and also introduce a casual attention module to the entropy estimator. We believe that the proposed technology has brought us one step closer to achieving high-perceptual-quality compression under ultra low bitrate conditions.

Acknowledgement

Computational resource of AI Bridging Cloud Infrastructure (ABCI) provided by National Institute of Advanced Industrial Science and Technology (AIST) was used.

References

[1] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. Generative adversarial networks for extreme learned image compression. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[2] Google AI. Overview of the open images challenge 2018. https://storage.googleapis.com/openimages/web/challenge.html, 2018.

[3] Hiroaki Akutsu and Takahiro Naruko. End-to-end learned roi image compression. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[4] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. *CoRR*, abs/1611.01704, 2016.

[5] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. *CoRR*, abs/1712.09763, 2017.

[6] Rich Franzen. Kodak lossless true color image suite. http://r0k.us/graphics/kodak/, 1999.

[7] Jun-Hyuk Kim, Jun-Ho Choi, Manri Cheon, and Jong-Seok Lee. RAM: residual attention module for single image super-resolution. *CoRR*, abs/1811.12043, 2018.

[8] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[9] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang. Learning convolutional networks for content-weighted image compression. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3214–3223, June 2018.

[10] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4394–4402, 2018.

[11] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Thirty-second Conference on Neural Information Processing Systems, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 10794–10803, 2018.

[12] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4790–4798. Curran Associates, Inc., 2016.

[13] Z Wang, Eero Simoncelli, and Alan Bovik. Multiscale structural similarity for image quality assessment. In *Conference Record of the Asilomar Conference on Signals, Systems and Computers*, 2003.

[14] Zhisheng Zhong, Hiroaki Akutsu, and Kiyoharu Aizawa. Channel-level variable quantization network for deep image compression. 2020 (under review).