A GAN-based Tunable Image Compression System

Lirong Wu
Zhejiang University
wulirong@zju.edu.cn

Kejie Huang
Zhejiang University
huangkejie@zju.edu.cn

Haibin Shen
Zhejiang University
shen_hb@zju.edu.cn

Abstract

The method of importance map has been widely adopted in DNN-based lossy image compression to achieve bit allocation according to the importance of image contents. However, insufficient allocation of bits in non-important regions often leads to severe distortion at low bpp (bits per pixel), which hampers the development of efficient content-weighted image compression systems. This paper rethinks content-based compression by using Generative Adversarial Network (GAN) to reconstruct the non-important regions. Moreover, multiscale pyramid decomposition is applied to both the encoder and the discriminator to achieve global compression of high-resolution images. A tunable compression scheme is also proposed in this paper to compress an image to any specific compression ratio without retraining the model. The experimental results show that our proposed method improves MS-SSIM by more than 10.3% compared to the recently reported GAN-based method [3] to achieve the same low bpp (0.05) on the Kodak dataset.

1. Introduction

Efficient image compression is significant for the storage, transmission, and processing of image information. At present, there are two types of image compression: lossy compression and lossless compression. The key point to lossy compression is to find a balance between the compression ratio and the distortion to guarantee the image quality at low *bpp* [6, 23]. Recently, lossy compression based on Deep Neural Networks (DNNs) is under focused development [2–4, 8, 18, 20]. The method of importance map has been widely adopted in DNN-based lossy image compression to achieve bit allocation according to the importance of image contents [13, 15]. However, its compression performance often dramatically drops at low *bpp*. In addition, there seem to be few tunable DNN-based image compression methods allowing an image to be compressed to any specific *bpp* without retraining the model.

In this paper, a novel GAN-based tunable image compression system aiming at low *bpp* is proposed to reconstruct the non-important regions of the image to compensate for the severe distortion caused by the insufficient allocation of bits in those non-important regions. The proposed system has been tested on the Kodak, ImageNet and Cityspace datasets. The experimental results show that our proposed scheme outperforms the-state-of-art schemes when *bpp* is smaller than 0.2. For example, our method achieves 10.3% higher MS-SSIM than [3] at low *bpp* (0.05) on the Kodak dataset. Moreover, our method can compress images to specified compression ratios without retraining the model. In contrast, the compression ratio of an image is unchangeable in [14, 15] because the importance map is deterministic for a given network structure. Therefore, they have to modify and retrain the model to generate new importance maps. Our contributions are listed as follows:

- We rethink content-based image compression under the GAN setting to reconstruct the non-important regions. We find that insufficient allocation of bits in non-important regions greatly limits the performance of content-based compression algorithms at low *bpp*.
- Unlike other methods using multiple complex networks to generate semantic maps and masks [3], we design a simple network (Masking) to identify the important regions of the image and generate the importance map to guide the allocation of bits.
- Different from [20], we use the multiscale structure not only in the encoder but also in the discriminator. The symmetrical multiscale structure makes it more adaptable to different sizes of objects at both the encoding end and the decoding end.
- We introduce tunability into our system. Unlike [14, 15], we achieve different compression ratios through an user-defined parameter n without retraining the model.

The rest of the paper is organized as follows: In Section 2, some common image compression algorithms and techniques are briefly reviewed. Section 3 describes the entire architecture and loss function of our model. Section 4

presents our experimental results and comparison with other methods. Section 5 analyzes and summarizes our results and Sectio 6 draws the conclusion.

2. Related Work

Recently, image compression based on deep learning has been a hot research topic. Up to now, data autoencoder [2, 4, 5, 16, 25, 30] and Recurrent Neural Networks (RNNs) [26, 27] are the two widely used models in the image compression architecture. Early works using block compression decompose the image into blocks, which are then compressed and composited [14, 15]. Recently, global compression of the entire high-resolution image is attracting more and more attention [3, 20, 27, 29].

GAN has been hailed as one of the greatest achievements in the field of deep learning in recent years. The idea is to construct a generator and a discriminator [10]. The training purpose of the discriminator $D(\cdot)$ is to maximize its discriminative accuracy, and the training goal of the generator $G(\cdot)$ is to improve the authenticity of its reconstructed image as much as possible. In the training process, GAN adopts an alternating optimization method, and its objective function can be expressed by the following formula:

$$\min_{G}\min_{D}\ \mathbb{E}[logD(x)] + \mathbb{E}\big[log(1 - D(G(x)))\big] \quad (1)$$

With the emergence of all kinds of variants such as conditional GAN [17] and CycleGAN [32], GANs have been widely applied in the field of computer vision [3, 20, 31, 32]. In the beginning, GAN is difficult to generate high-resolution images, which greatly limits its application. Recently, GAN is under intense development, and the high-resolution images can be synthesized by GAN [7, 29]. For example, TC Wang et al. present a method for synthesizing 2048×1024px photo-realistic images from semantic label maps using conditional GAN in [29]. Therefore, GAN is adopted to achieve global image compression [3].

At present, some GAN-based image compression methods have been proposed [3, 8, 20, 22], but neither of them considers the influence of image content importance on bit allocation, which limits GAN's effect on image compression. The method proposed by Santurkar et al. is to train thumbnail images to get an efficient generator, but the information contents of the thumbnail image are so low that GAN can't play much of a role [22]. The closest work to us is [3], which trains a GAN-based system to achieve the compression of images. However, this scheme has several weaknesses, which limit it in practical applications. As shown in Fig. 1, their method requires multiple complex semantic segmentation network and feature extraction network to generate semantic maps and masks. Instead, we just design a simple network (Masking) to identify the important regions of the image and generate the importance map to guide the allocation of bits. Secondly, due to the complexity of their entire architecture, their codec efficiency is so low that it can't meet the needs of practical applications at all. Moreover, changing the compression ratio has to reset parameters and retrain the model in their framework.

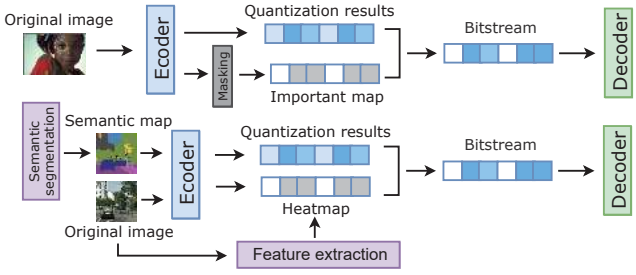


Figure 1. Top: our method, bottom: other method [3]

3. Model

3.1. Architecture

Our image compression system is composed of six parts: encoder, quantizer, masker, entropy encoder, decoder, and discriminator. The entire architecture is shown in Fig. 2. For a given image $x \in \mathrm{X}$, the encoder converts it into a compact code matrix $\omega = E(x)$, by multiscale convolution operations. The masker takes ω as the input and generates an importance matrix $\widehat{m} = M(\omega)$ through a simple convolutional network [14, 15] to guide the bit-allocation. The quantizer $Q(\cdot)$ quantizes ω by using a nearest neighbor principle [2, 15, 25] and outputs $\widehat{q} = Q(\omega)$. The output of the quantizer and the masker are multiplied to gain the content-based image compression result, denoted as $z = \widehat{m} \cdot \widehat{q}$. The masker here can be understood as obscuring the non-important regions in the image and allocating more bits to the important regions. Entropy encoder is applied to the system to remove the data redundancy and outputs $\widehat{h} = H(z)$. The decoder $G(\cdot)$, which is also named as generator, is the inverse of the encoder and generates the reconstructed image $\widehat{x} = G(z)$. The discriminator $D(\cdot)$ is an important part of the GAN, which improves the compression performance through alternating training [3] with the generator.

The six components of the image compression system will be introduced in the rest of the sections in detail.

Encoder

In our image compression system, a fully convolutional neural network is used as the encoder, which consists of a crossover stack of several convolutional layers and residual blocks. To improve the compression performance of high-resolution images, we adopt the pyramidal decomposition scheme shown in Fig. 3 to our model.

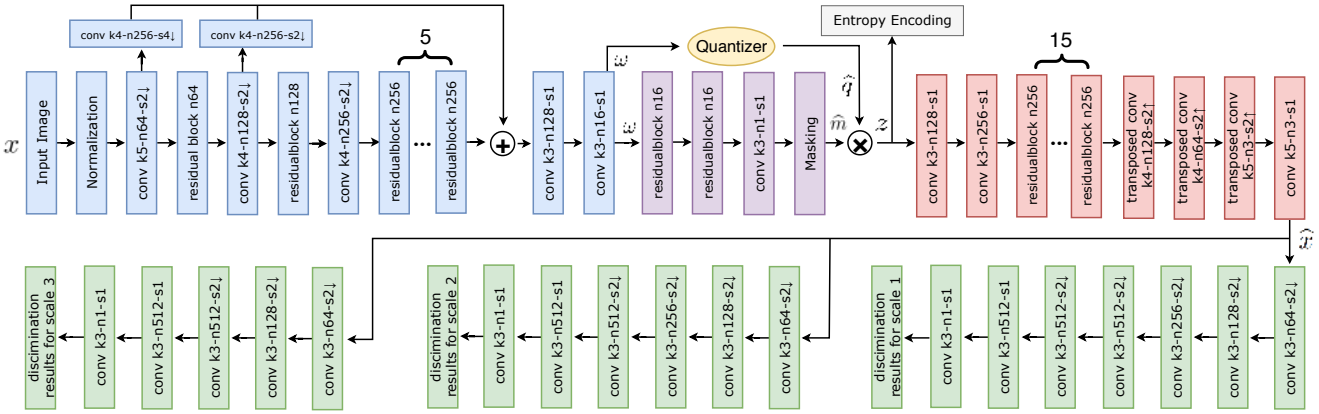


Figure 2. Illustration of the GAN-based tunable image compression system. In the figure, blue, purple, yellow, gray, red, and green blocks represent encoder, masker, quantizer, encoder, decoder, and discriminator, respectively. It is noted that "k5-n64-s2↓" represents a convolution layer with 64 filters of size 5 × 5 and a stride of 2. Each residual block has a uniform structure composed of two convolutional layers followed by a batch normalization [12] and a *ReLU* [9]. Masking is an operation that extends the importance map to importance matrix according to Eq.(3)

Let x_m denotes the input of the scale m layer, so x_1 denotes the original input image. $E_m(x_m)$ represents the output of the scale m layer. In our paper, we set m to 1, 2, and 3 sequentially and execute encoding individually for each scale. The results of each scale are weighted and summed to produce an output $E(x) = \alpha_1 E_1(x_1) + \alpha_2 E_2(x_2) + \alpha_3 E_3(x_3)$. Finally, $E(x)$ is convoluted with two convolutional layers to get the output of the encoder ω with the dimension of $\frac{H}{8} \times \frac{W}{8} \times K$. According to [3], different K produces different compression effects, which is a trade-off between the compression ratio and the distortion.

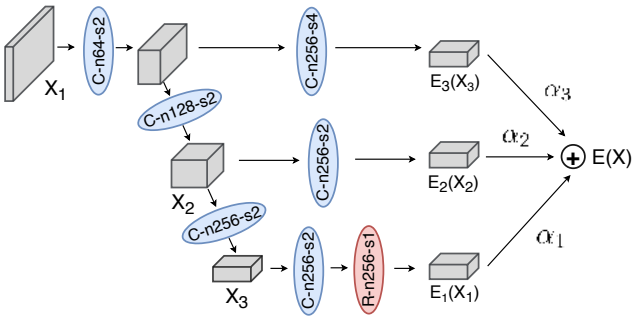


Figure 3. Illustration of the encoder's pyramidal decomposition structure with 3 scales. It's noted that "C-n16-s2" represents a convolutional layer with 16 filters and a stride of 2 and "R-n256-s1" represents a residual block with 256 filters and a stride of 1.

Masker

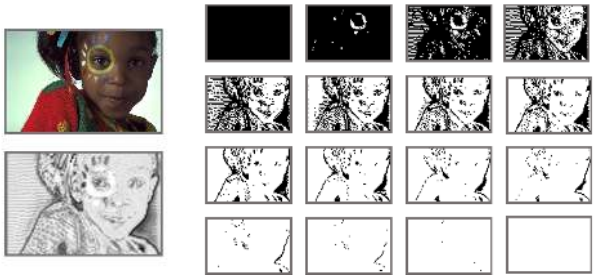
In an image, we tend to be interested only in some regions, which provides leeway for further improvement in compression ratio. For example, for the portrait shown in Fig. 4, we are only interested in the face and body regions, which are called the important regions. The natural idea is that more bits are allocated to the important regions and fewer bits are allocated to the non-important regions. The bit allocation according to the importance of image contents is achieved by constructing a masker.

The output of the encoder ω is used as the input of the masker, which is convoluted with two residual blocks. Each residual block has 256 filters. The kernel size and stride length of each filter are 3 × 3 and 1, respectively. These residual blocks are followed by a convolutional layer with 1 filter. The size of the output matrix y is $\frac{H}{8} \times \frac{W}{8} \times 1$. As shown in Fig. 5, a *Sigmoid* activation is used to map the data y to the range [0,1] to get an importance map m. However, the data after *Sigmoid* may converge to 0 or 1, resulting in the vanishing of the importance feature of m. To avoid this issue, we normalize the data in the matrix y before the activation. The formula of the normalization procedure is specified as

$$\widehat{y}_{i,j} = \frac{y_{i,j} - \overline{\mu}}{\overline{\sigma}}, \; m_{i,j} = tf.nn.sigmoid(\widehat{y}_{i,j}) \qquad (2)$$

where $y_{i,j}$ represents the data value of the i-th row and the j-th column in the matrix y and $\widehat{y}_{i,j}$ represents the data value of the i-th row and the j-th column in the matrix $\widehat{y}$. $\overline{\mu}$ and $\overline{\sigma}$ are the mean and variance of the matrix y, respectively. As mentioned earlier, we aim to design an tunable image compression system to compress the image to any *bpp* without retraining the model. So here we replace $\overline{\mu}$ with $\overline{\mu} + n$ in the Eq.(2), where n is a random number within the range of [-2,2], and it is reassigned before each training batch.

The importance map m is extended by the formula Eq.(3) to the importance matrix $\widehat{m}$, as shown in Fig. 5.



(a) (b)

Figure 4. (a) The upper image is the original image x, and the lower image is the importance map m (b) Images of each channel of the importance matrix $\widehat{m}$ (take $K = 16$ ans $n = 0$ as an example). The size of the $\widehat{m}$ is $\frac{H}{8} \times \frac{W}{8} \times K$. From left to right, top to bottom, the channel of the matrix $\widehat{m}$ gradually rises. The black regions represent importance regions.

$$\widehat{m}_{i,j,k} = \begin{cases} 0, & \text{if } m_{i,j} < \frac{k-1}{K} \\ 1, & \text{if } m_{i,j} \geq \frac{k-1}{K} \end{cases} \quad k = 1, \ldots, K \qquad (3)$$

where $m_{i,j}$ represents the data value of the i-th row and the j-th column in the matrix m, and $\widehat{m}_{i,j,k}$ represents the data value of the i-th row, the j-th column and the k-th channel in the matrix $\widehat{m}$. Taking the image x in Fig. 5 as an example, the original image, the importance map and the images of each channel in the importance matrix are shown in Fig. 4. As the channel rises, the bits are mainly allocated to the face and body regions, and hardly in the background, which contributes to the improvement of the compression ratio. However, the image may be severely distorted at low *bpp* due to the insufficient bit allocation in the background regions. To address this issue, we reconstruct the non-important regions of the image based on GAN to improve the performance.

Quantizer

The selection of quantization bit is very important to the quantizer. Appropriate quantization bit not only improves the compression ratio, but also reduces the distortion. We set $C_L = \{0, 1, 2, ..., 2^L\text{-}1\}$, and there are plenty of methods to quantize the input to a number in C_L. Here we use the nearest neighbor quantization method [2,15,25] to compute:

$$\widehat{q} = Q(\omega) = \arg\min_j |\omega - c_j| \qquad (4)$$

where $c_j = j$, and $j \in C_L = \{0, 1, 2, ..., 2^L\text{-}1\}$.

Entropy encoder

As shown in Fig. 2, in our method, images are reconstructed directly from the bitstream z instead of the quantization results $\widehat{q}$ and the importance matrix $\widehat{m}$, which is exactly what we differ from other methods. The importance matrix $\widehat{m}$ indicates the non-mask bits (in gray) and the mask bits (in white) in $\widehat{q}$. As shown in Fig 5, the code matrix z obtained by multiplying the results of masker and quantizer leaves only the non-mask bits. Then we encode each channel of z from the bottom up in a row-by-row manner. Here we can only encode the non-mask bits. A large number of mask bits (data 0) at the top of each channel in z can be encoded as a termination code instead of being encoded one by one. When the mask bits are much more than the non-mask bits, our compression ratio can reach very low *bpp*. Since only a few non-mask bits at the bottom of each channel need to be encoded, we use simple Huffman coding for entropy coding in this work [11].

Decoder

The decoder is the inverse of the encoder, and its function is to generate images with minimal distortion from the

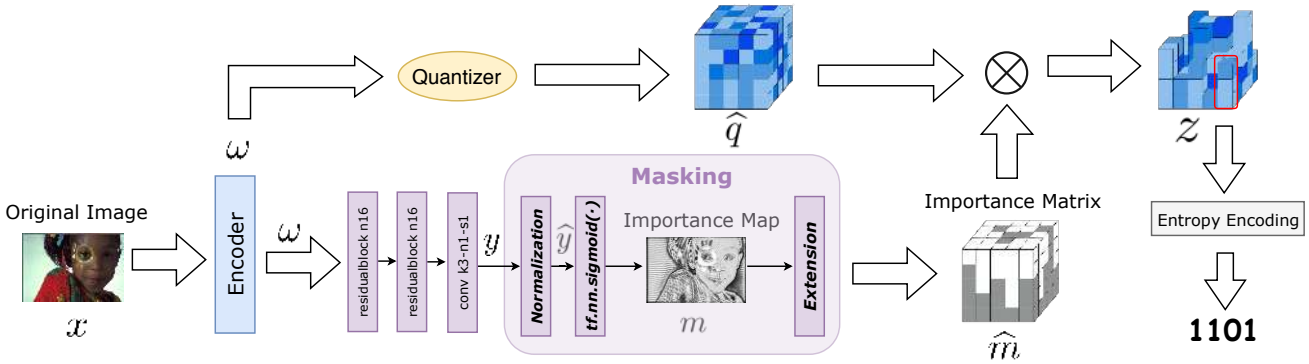


Figure 5. Illustration of how to generate an importance map m and an importance matrix $\widehat{m}$ from the original image x and use the importance matrix $\widehat{m}$ to guide the bit-allocation of the quantization result $\widehat{q}$. The colors of cubes in $\widehat{q}$ are from light to deep, respectively, and the quantized values are from 0 to 3. The gray and white cubes in $\widehat{m}$ represent 1 and 0, respectively. The light purple box corresponds to the **Masking** block in Fig. 2. *Normalization*, *tf.nn.sigmoid(·)* and *Extension* correspond to the preprocessing of y and the extension of the m. If the Huffman encoding is specified, that is 0-1, 1-01, 2-001 and 3-0001, then the circled part of z is quantized to 1101.

compression code matrix z. A good decoder should make restructured images as similar as possible to the original images in terms of texture, color, and so on. The decoder is composed of a stack of 3 convolution layers, 15 residual blocks, and 3 transposed convolution layers. First of all, the input z is convoluted with 128 filters of size 3×3 and stride 1. After that, the obtained feature maps are convoluted with 256 filters of size 3×3 and stride 1, followed by 15 residual blocks. Similar to [14], these residual blocks are identical in the proposed model, consisting of two convolutional layers with 256 filters. Finally, the output of the last stage residual block is passed through 3 transposed convolutional layers to generate a reconstructed image $\widehat{x}$.

In fact, the decoder is also the generator in our GAN-based system. It improves its performance during the alternating training with the discriminator and generates images that the discriminator cannot identify the authenticity.

Discriminator

The discriminator $D(\cdot)$ is able to identify the authenticity of the input image, i.e., whether it is the original image or the reconstructed image. As an important part of GAN, the discriminator $D(\cdot)$ is trained in parallel with the generator $G(\cdot)$ [3, 20] to improve the performance of generating images. In this paper, we continue to use the idea of pyramidal decomposition to design a multiscale discriminator. The motivation for adopting a multiscale architecture is to minimize the distortion at each scale separately. For example, artifacts such as noise and blurriness are more easily found and eliminated at shallower scales, but the differences of the structure and the color of the image are usually found at deeper scales. Here we assume that $\widehat{x}$ is the input of the discriminator $D(\cdot)$, and the input $\widehat{x}_m$ of the corresponding scale m is obtained by the average pooling layer with a stride of 2. The input of each scale passes through a convolutional network $D_m(\cdot)$ to produce an output $D_m(\widehat{x}_m)$. Each convolutional layer of the networks is followed by *Leaky ReLU* instead of *ReLU* as the activation [29].

3.2. Loss function

In the previous sections, we have designed an image compression system based on GAN. Now we train the model on a batch of B, that is X_B=$\{X^{(1)}$,$X^{(2)}$,$\cdots$,$X^{(B)}\}$, containing high-resolution images. The loss function of our model is composed of the following two parts.

Adversarial Loss

We design our compression system based on GAN. The generator $G(\cdot)$ is trained in parallel with the discriminator $D(\cdot)$. We call this part of the loss as adversarial loss, which is composed of the losses from generator $G(\cdot)$ and discriminator $D(\cdot)$. The adversarial loss is defined as follows:

$$\mathcal{L}_A = \sum_{i=1}^{m} \beta_i \Big\{ \mathbb{E}[logD_i(x)] + \mathbb{E}\big[log(1-D_i(G(x)))\big] \Big\} \quad (5)$$

where x is the original image, m is the scale of discriminator, and β_i is the weighting factor for scale i.

Distortion Loss

The distortion loss measures the distortion of the original image x and reconstructed image $\widehat{x}$. The purpose of the training is to minimize the following loss:

$$\mathcal{L}_D = E[d(x, \widehat{x})] \quad (6)$$

where $d(\cdot)$ is a function to measure the similarity of the original image x and reconstructed image $\widehat{x}$. In this paper, the Mean Square Error (MSE) is used in the distortion loss.

Overall Loss

There is a constraint relationship between the above two losses. For example, increasing the adversarial loss may produce more generated contents in the reconstructed image, resulting in an increase in the distortion loss. Therefore, in the training process, we should consider the above two losses comprehensively. Since we train the model on the batch X_B of size B, we need to consider the loss on the entire batch. The overall loss function is expressed as

$$\begin{aligned} \mathcal{L}_{G,D,E,B} = \frac{1}{B} \sum_{j=1}^{B} \Big\langle \eta \sum_{i=1}^{m} \beta_i \Big\{ \mathbb{E}\big[log(1-D_i(G(x^j)))\big] \\ + \mathbb{E}[logD_i(x^j)] \Big\} + \kappa E\big[d(x^j, \widehat{x}^j)\big] \Big\rangle \end{aligned} \quad (9)$$

The training purpose is to minimize the overall loss.

$$\min_{G,E,B} \min_{D} \mathcal{L}_{G,D,E,B} \quad (7)$$

4. Experiments and Results

Our GAN-based tunable image compression system is trained on a subset of 15000 images in the ImageNet database [21]. All images are scaled to 768 × 512, and every eight images are packed into a batch. Then, we test the model on the Kodak dataset [1] which is specifically designed to test the performance of lossy image compression. The compression ratio of the image is evaluated by *bpp*, which is the average number of bits required per pixel to store the compressed result. The distortion between the original and restructured image is commonly measured by MSE [2–4,15,25], PSNR, MS-SSIM [15,20,27]. Compared

with MSE, PSNR and MS-SSIM are used in this paper because they are more consistent with the actual perception of human vision [19]. In addition, we also compare the performance of our model on several different datasets.

In the rest of this section, we first introduce the parameter settings of our model, then compare the performance and visual effects of different compression methods and datasets. Besides, we perform an ablation experiment to show the impact of the importance matrix, entropy coding, GAN. Finally, we analyze the compression tunability of our system.

parameter settings

Firstly, we set the weights α_1, α_2, and α_3 of 3 scales in the encoder to 1/2, 1/4, and 1/4, respectively. Similarly, the weights β_1, β_2, and β_3 of 3 scales in the decoder are set to 1/2, 1/4, and 1/4, respectively. The weights η and κ of two loss components of the overall loss are set to 1 and 16, respectively. In addition, we set the batch parameter B to 8, which is to train the model using 8 images as a batch. Moreover, we set the quantization parameter L to 2, which means $C_L = \{0, 1, 2, 3\}$. In this work, if not specifically mentioned, let $K = 16$. During the training process, the model is iteratively trained 128 times on the dataset. The initial learning rate is set to 2×10^{-3}, and after 64 iterations, the learning rate is changed to 2×10^{-4}.

Comparison of different methods

Firstly, We compare the MS-SSIM performance of our tunable and non-tunable method with some conventional methods such as JPEG [28], JPEG2000 [24], and BPG on the Kodak dataset [1]. We divide our compression system into two cases: tunable and non-tunable, which can be achieved by setting n to a random value and a fixed value during the training process, respectively. In addition, some DNN-based methods, such as [15, 20, 25], are also included in the comparison.

As shown in Fig. 6, our method outperforms JPEG, JPEG 2000, BPG and the method proposed by Theis et al [25] at a wide range of scale. At high *bpp*, the performance of our method is close to that of Mentzer et al. [15] and Rippel & Bourdev [20], but at low *bpp*, our performance is much better than their methods. For example, compared with the method proposed by Mentzer et al., the *bpp* of our tunable and non-tunable models is reduced by 30.1% and 39.3% when MS-SSIM is 0.95, respectively.

Next, we further compare the PSNR performance of different methods based on the work of [13]. We compare our non-tunable method with JPEG 2000 and BPG as well as the methods proposed in [5, 14, 25]. In addition, the latest content-adaptive method proposed by Jooyoung et al. is also included in the comparison. As shown in Fig. 6, at low *bpp*, the PSNR performance of our method is still superior to other methods. At high *bpp* (*bpp* $\geqslant$ 0.5), our method will be slightly worse than [13], because at this time even the non-importance regions have been allocated enough bits, more and more contents generated by the GAN result in a decrease in PSNR performance. However, as we have always emphasized, our method focuses on the performance at extreme low *bpp*, and it's acceptable to have a general performance at high *bpp*. The performance on different datasets is available in the supplementary materials.

Ablation experiments

The use of GAN in our proposed image compression system is to eliminate the distortion caused by insufficient bit allocation to non-important regions rather than generate new image contents. In our method, sufficient bits are allocated to the important regions, which can be reconstructed realistically, and GAN has less impact on them. Under such circumstances, the main basis for the discriminator to discriminate is the non-important regions. So to confuse the discriminator, the generator will focus on reconstructing non-important regions. The role of the importance matrix is to guide the allocation of bits and improve the representative efficiency of the bits [3]. The function of entropy coding is to further reduce data redundancy by

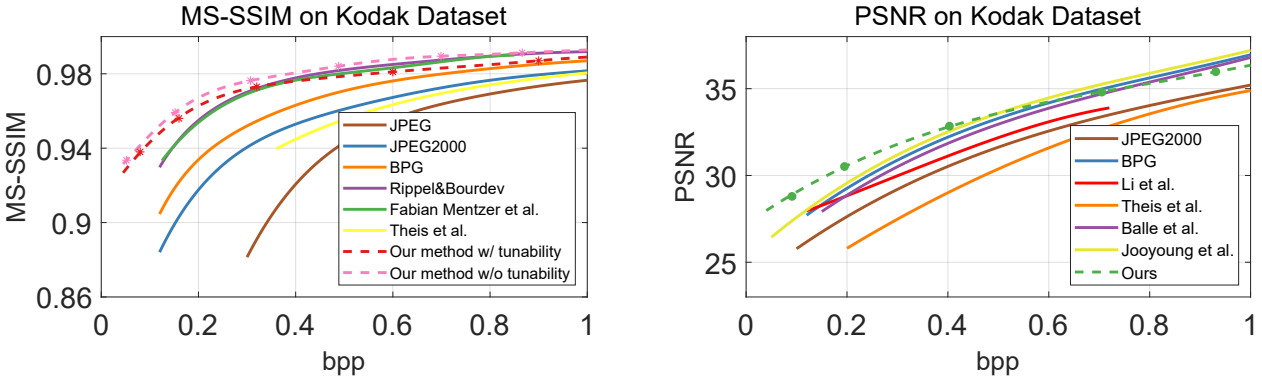


Figure 6. Comparison of compression performance by different methods measured by MS-SSIM and PSNR.

exploiting the specificity of data distribution in the importance matrix. We design the following five models according to whether the presence of GAN, masker and entropy coding in the architecture: (1) full model; (2) model without GAN; (3) model without masker; (4) model without entropy coding; (5)model without masker and entropy coding; As shown in Fig. 7, under the same compression performance, (1) has the best performance while (5) has the worst performance. When MI-SSIM is 0.96, (2), (3), (4), and (5) has 15.8%, 89.3%, 31.6%, and 116.3% higher *bpp* than (1), respectively. The performance of the model with GAN performs better than that of the model without GAN at low *bpp*. However, the improvement is gradually diminished with an increase of *bpp*. At high *bpp*, the introduction of GAN may even slightly impair compression performance. This shows that GAN's help with image compression is more pronounced at low *bpp*, which is not mentioned in other GAN-based methods. In the compression task, we usually want the *bpp* to be as small as possible, so the introduction of GAN can help solve the bottleneck limiting of the performance improvement at low *bpp*.

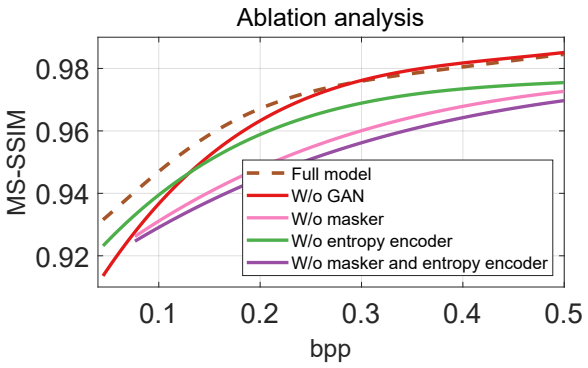


Figure 7. Illustration of the results of the ablation experiment.

Tunability analysis

Our system has the tunable characteristic, which means we can reassign the user-defined parameter n in the masker to achieve different compression ratios without retraining the model. However, in the methods like [14, 15], for an image, one type of network structure corresponds to one unique importance map. Therefore, only by modifying and retraining the model, it is possible to obtain different importance map, then achieve different compression ratios.

The compression ratio of the image is determined by the parameter *n*, which is an intuitive and simple dependency. However, it should be noted that *n* and *bpp* may not be in a strictly linear relationship. In the process of testing, different *n* is used to get its corresponding *bpp*. The data are fitted by Least Squares Method (LSM) to gain the tunability characteristic curve, as shown in Fig. 8. The image can be compressed to any specific *bpp* within the range of [0.05, 0.4] as long as we set *n* to the corresponding value. For example, if we want to compress an image to 0.384 *bpp*, then by looking up the figure and setting n to -1.32, we can get a compression ratio around 0.384 *bpp*. The red dots in Fig. 8 are the test results of images on the Kodak dataset.

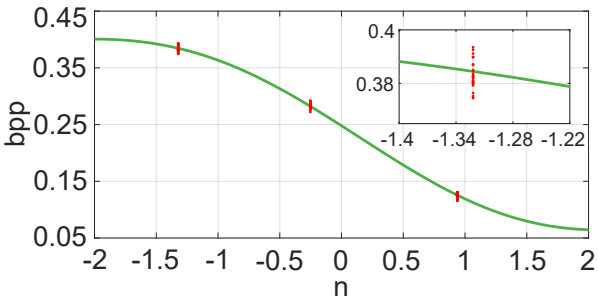


Figure 8. Tunability characteristic curve of the image compression system. The red dots represent the different compression ratios tested on the Kodak dataset at a fixed n.

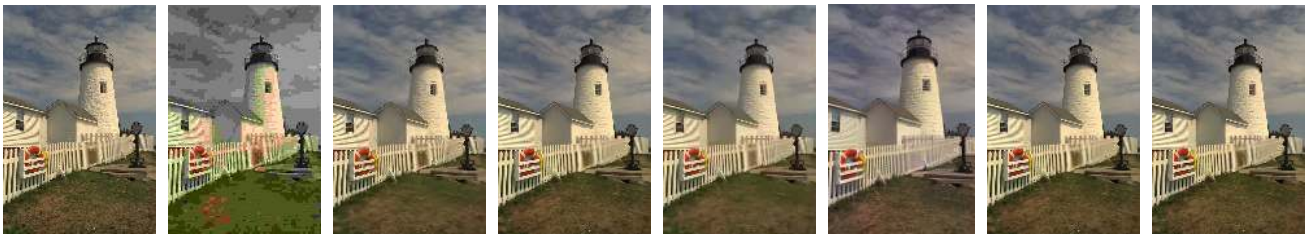
Comparison of visual effects

In Fig. 9, we compare our methods with JPEG, JPEG 2000, BPG as well as the methods of Fabian Mentzer et al. [15] and Rippel & Bourdev [20] visually. As can be seen from Fig. 9, conventional image compression methods such as JPEG, JPEG 2000 and BPG inevitably produce blurring, ringing, etc. [3], which can seriously affect the human visual experience. Though the methods of Fabian Mentzer et al. [15] and Rippel & Bourdev [20] are very good at detail processing, they fail to show the structure and color of the image well. In contrast, our method overcomes the above flaws, and some important colors and textures are well-retained and more visually pleasing due to the bit-allocation based on the image contents.

In Fig. 10, we compare our non-tunable method with the most advanced GAN-based method at low *bpp*. Compared with [3], since we introduce the important matrix into our system, the details of the image, such as the window of the house, the lock on the door, the holes in the woman's hat and the fuselage and paddles of the aircraft, are well preserved. Besides, due to the use of GAN, the non-importance regions of the image are also very harmonious, without severe distortion resulted from the lack of bits. In term of MS-SSIM, since [3] is too dependent on GAN, their MS-SSIM is only 83.9% at 0.05 *bpp*. In contrast, our MS-SSIM is 10.3% higher than theirs. For more visual comparisons, please refer to the supplementary materials.

5. Discussion

In our architecture, we design the multiscale encoder and discriminator based on the idea of pyramidal decomposition, introduce importance map for bit allocation, and



(a) Original (b) Jpeg (c) Jpeg2000 (d) Bpg (e) Mentzer et al. (f) Rippel et al. (g) Tunable (h) Not tunable

Figure 9. Illustration of the original image and the reconstructed images produced by conventional and DNN-based compression methods. From left to right, the *bpp* of each method is 24 *bpp*, 0.123 *bpp*, 0.125 *bpp*, 0.108 *bpp*, 0.128 *bpp*, 0.093 *bpp*, 0.116 *bpp*, 0.103 *bpp*.

(a) *bpp* / MS-SSIM: Ours 0.039 / 0.927, Agustsson et al. 0.030 / 0.824

(b) *bpp* / MS-SSIM: Ours 0.063 / 0.906, Agustsson et al. 0.069 / 0.795

(c) *bpp* / MS-SSIM: Ours 0.058 / 0.921, Agustsson et al. 0.065 / 0.845

(d) *bpp* / MS-SSIM: Ours 0.040 / 0.937, Agustsson et al. 0.034 / 0.844

Figure 10. Illustration of comparison with the state of the art GAN-based method. From left to right: Original, Ours, Agustsson et al.

further compress data by entropy coding. As for the training approach, we introduce two losses, all of which are weighted and summed to get the overall loss function and use global compression of high-resolution images instead of block compression. At the same time, we introduce GAN to reconstruct non-important regions of the image to solve the distortion caused by insufficient bit allocation to non-important regions.

The experimental results show that our method outperforms the-state-of-art content-based and GAN-based methods when *bpp* is smaller than 0.2. At low *bpp*, the existence of GAN has a more significant impact on performance improvement because the insufficient bit allocation in the non-importance regions often occurs in the case of low *bpp*. In terms of MS-SSIM and PSNR, our method is superior to conventional compression algorithms such as JPEG, JPEG2000 and BPG, and also outperforms the state-of-the-art DNN-based compression methods at low *bpp*. Visually, our approach solves the flaws of conventional algorithms such as ringing, blurring, etc., and can better preserve the texture, color, and other details of the image. In addition, as shown in Fig. 8, our system has the tunable characteristic, and within a certain range, the compression ratio of any *bpp* can be achieved through an user-defined parameter n without retraining the model. On the Kodak dataset, to achieve MI-SSIM of 0.95, the average time to encode and decode the image is 21 ms and 29 ms, running on the GeForce GTX 1080 Ti.

6. Conclusions

In this paper, we have proposed a GAN-based tunable lossy image compression system. In the proposed system, GAN is trained to reconstruct the non-important regions of the image and thus reduce the distortion caused by the insufficient bit allocation to those non-important regions. More importantly, the idea of tunability has been applied to the DNN-based image compression systems. Our compression system has the tunability characteristic, which means we can compress an image to a specific compression ratio without retraining the model.

References

[1] Kodak photocd dataset.

[2] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*, pages 1141–1151, 2017.

[3] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. Van Gool. Generative adversarial networks for extreme learned image compression. *arXiv preprint arXiv:1804.02958*, 2018.

[4] J. Ballé, V. Laparra, and E. P. Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

[5] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

[6] Y. Blau and T. Michaeli. Rethinking lossy compression: The rate-distortion-perception tradeoff. *arXiv preprint arXiv:1901.07821*, 2019.

[7] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[8] L. Galteri, L. Seidenari, M. Bertini, and A. Del Bimbo. Deep generative adversarial compression artifact removal. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4826–4835, 2017.

[9] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[11] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[13] J. Lee, S. Cho, and S.-K. Beack. Context-adaptive entropy model for end-to-end optimized image compression. *arXiv preprint arXiv:1809.10452*, 2018.

[14] M. Li, W. Zuo, S. Gu, D. Zhao, and D. 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.

[15] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool. Conditional probability models for deep image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4394–4402, 2018.

[16] D. Minnen, J. Ballé, and G. D. Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10771–10780, 2018.

[17] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[18] Y. Patel, S. Appalaraju, and R. Manmatha. Deep perceptual compression. *arXiv preprint arXiv:1907.08310*, 2019.

[19] Y. Patel, S. Appalaraju, and R. Manmatha. Human perceptual evaluations for image compression. *arXiv preprint arXiv:1908.04187*, 2019.

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

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[22] S. Santurkar, D. Budden, and N. Shavit. Generative compression. In *2018 Picture Coding Symposium (PCS)*, pages 258–262. IEEE, 2018.

[23] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[24] A. Skodras, C. Christopoulos, and T. Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001.

[25] L. Theis, W. Shi, A. Cunningham, and F. Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

[26] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar. Variable rate image compression with recurrent neural networks. *arXiv preprint arXiv:1511.06085*, 2015.

[27] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell. Full resolution image compression with recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5306–5314, 2017.

[28] G. K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[29] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.

[30] L. Zhou, C. Cai, Y. Gao, S. Su, and J. Wu. Variational autoencoder for low bit-rate image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2617–2620, 2018.

[31] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[32] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.