

# Channel-Level Variable Quantization Network for Deep Image Compression

Zhisheng Zhong<sup>1</sup>, Hiroaki Akutsu<sup>2</sup> and Kiyoharu Aizawa<sup>1</sup>

<sup>1</sup>The University of Tokyo, Japan

<sup>2</sup>Hitachi Ltd, Japan

zhisheng@hal.t.u-tokyo.ac.jp, hiroaki.akutsu.cs@hitachi.com, aizawa@hal.t.u-tokyo.ac.jp

## Abstract

Deep image compression systems mainly contain four components: encoder, quantizer, entropy model, and decoder. To optimize these four components, a joint rate-distortion framework was proposed, and many deep neural network-based methods achieved great success in image compression. However, almost all convolutional neural network-based methods treat channel-wise feature maps equally, reducing the flexibility in handling different types of information. In this paper, we propose a channel-level variable quantization network to dynamically allocate more bitrates for significant channels and withdraw bitrates for negligible channels. Specifically, we propose a variable quantization controller. It consists of two key components: the channel importance module, which can dynamically learn the importance of channels during training, and the splitting-merging module, which can allocate different bitrates for different channels. We also formulate the quantizer into a Gaussian mixture model manner. Quantitative and qualitative experiments verify the effectiveness of the proposed model and demonstrate that our method achieves superior performance and can produce much better visual reconstructions<sup>1</sup>.

## 1 Introduction

Since the development of the internet, increasingly more high-definition digital media data has overwhelmed our daily life. Image compression refers to the task of representing images using as little storage as possible and is an essential topic in computer vision and image processing.

The typical image compression codecs, e.g., JPEG [Wallace, 1992] and JPEG 2000 [Skodras *et al.*, 2001], generally use some transformations such as discrete cosine transform (DCT) and discrete wavelet transform (DWT), which are mathematically well-defined. These compression methods do not fully utilize the nature of data and may introduce visible artifacts such as ringing and blocking. In the last several years, deep learning has revolutionized versatile com-

puter vision tasks [Krizhevsky *et al.*, 2012; Dong *et al.*, 2014; He *et al.*, 2016]. Image compression based on deep learning, or deep image compression for brevity, has become a popular area of research, which can possibly explore the use of the nature of images beyond conventional compression methods.

A deep image compression system is similar to the conventional one, as it usually includes four components, i.e., encoder, quantizer, entropy model, and decoder, to form the final codec. To train a deep image compression system, a rate-distortion trade-off loss function:  $R + \lambda D$  was proposed in [Ballé *et al.*, 2017], where  $\lambda$  is the balanced hyper-parameter. The loss function includes two competing terms, i.e.,  $R$  measures the bitrate of the final compressed code, and  $D$  measures the distortion between the original and reconstructed images. Recently, to improve the performance of the deep image compression system further, researchers proposed many novel and effective derivatives for the above four components.

**En/Decoder.** The most popular architecture for en/decoder is based on convolutional neural network (CNN). E.g., [Ballé *et al.*, 2017] proposed a three convolutional layers en/decoder and generalized divisive normalization (GDN) for activation. [Li *et al.*, 2018] proposed a nine convolutional layers en/decoder with the residual block ([He *et al.*, 2016]). Google Inc presented three variants ([Toderici *et al.*, 2016; Toderici *et al.*, 2017; Johnston *et al.*, 2018]) of a recurrent neural network (RNN)-based en/decoder to compress progressive images and their residuals. [Agustsson *et al.*, 2019] proposed a generative adversarial network (GAN)-based en/decoder for extremely low bitrate image compression, which achieved better user study results.

**Quantizer.** In conventional codecs, the quantization operation is usually implemented by the round function. However, the gradient of the round function is almost always zero, which is highly unsuitable for deep compression. Thus, many differentiable quantization approaches were proposed by researchers. In [Toderici *et al.*, 2016], Bernoulli distribution noise was added to implement the map function from continuous values to the fixed set  $\{-1, +1\}$ . The importance map was proposed in [Li *et al.*, 2018] to address the spatial inconsistency coding length. Based on the K-means algorithm, the soft quantizer [Mentzer *et al.*, 2018] was proposed for the multi-bits quantization case. [Ballé *et al.*, 2017] proposed uniformed additive noise for infinite range quantization, whose quantization level is undetermined.

<sup>1</sup>Code address: <https://github.com/zzs1994/CVQN>

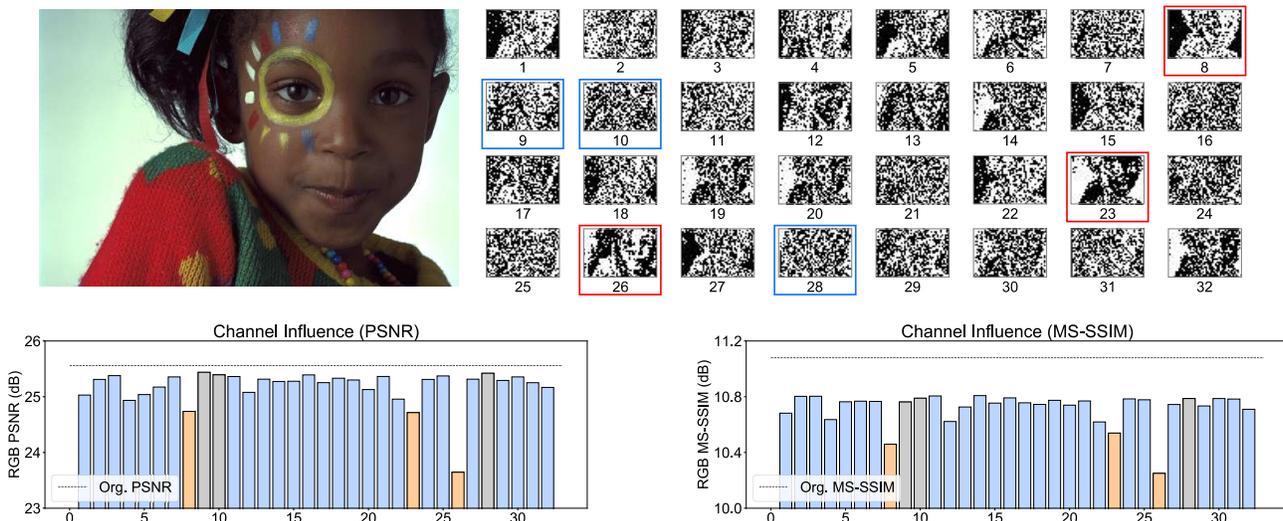


Figure 1: Illustration of channel influences. Top left: The original example image (*kodim15*) from the Kodak dataset. Top right: The visual results of the quantized feature map (channel by channel, 32 channels in total). Bottom left: PSNR loss of each channel. Bottom right: MS-SSIM loss of each channel in decibels:  $-10 \log_{10}(1 - \text{MS-SSIM})$ . *Best viewed on screen.*

**Entropy model.** To further compress the quantized code by through entropy coding methods, e.g., Huffman coding and arithmetic coding, the entropy model was proposed to regularize the redundancy among the quantization code. Almost all entropy arithmetic coding models are motivated by the context-based adaptive binary arithmetic coding (CABAC) framework. Specifically, in [Toderici *et al.*, 2017], they used PixelRNN [Van Oord *et al.*, 2016] and long short term memory (LSTM) architectures for entropy estimation. In [Mentzer *et al.*, 2018], they utilized a 3D convolutional model to generate the conditional probability of the quantized code. In [Ballé *et al.*, 2018; Minnen *et al.*, 2018; Lee *et al.*, 2019], they proposed a variational autoencoder (VAE) with a scale hyperprior to learn the context distribution, which achieves consequently achieving better compression results.

## 2 Channel Influences

All previous deep image compression systems view all channels as a unified whole and ignore the channel-level influences. However, useful information is unevenly distributed across channels. Channel redundancy and uneven distribution have been widely studied in the field of network compression [Luo *et al.*, 2017; Liu *et al.*, 2017; He *et al.*, 2017]. In this study, we utilize a toy example model to illustrate its feasibility in deep image compression. We use a simple encoder and quantizer to extract features and quantize them. The final quantized feature map has 32 channels. We allocate one bit for quantization, i.e., its quantization level is two. Evaluating on the Kodak dataset, this toy model yields an average MS-SSIM [Wang *et al.*, 2003] of 0.922 at an average rate of 0.112 bits per pixel (BPP). In the top part of Fig. 1, we present the visual results of the quantized feature map (channel by channel, 32 channels) by using *kodim15* from Kodak. The visual results indicate that Channel-8, 23, and 26 have similar content and profile (similar to low-frequency information) with the original image. By contrast, some visualizations, e.g., Channel-9, 10, and 28 appear disorganized and could not be

recognized (similar to high-frequency information). We also make quantitative comparisons. We conduct 32 experiments. In each experiment, we cut one relative channel (set its values to 0) of the quantized feature map to observe the influence of each channel on the final reconstruction results. The bottom of Fig. 1 depicts the PSNR loss of each channel on the left and the MS-SSIM loss of each channel on the right. Consistent with the analysis of visual results, Channel-8, 23, and 26 are significant for reconstruction, whereas Channel-9, 10, and 28 are negligible. Moreover, this phenomenon appears on all images of the dataset. Thus, the problem is as follows: Can we design a variable deep image compression system to ensure the allocation of more bits for important channels and the reduction of bitrate for negligible channels? In this paper, we propose a novel network to solve this issue.

The overall contributions of this study are three-fold:

- We analyze the channel influences in deep image compression. We propose a novel variable channel controller to effectively utilize channel diversity. To the best of our knowledge, we are the first to perform image compression in a channel-level manner.
- We propose a novel quantizer based on Gaussian mixture model (GMM). This novel quantizer has powerful representation and is a more generalized pattern for the existing finite quantizers.
- Extensive quantitative and qualitative experiments show that our method achieves superior performance over the state-of-the-art methods without a hyperprior VAE.

## 3 Approach

The framework of the proposed system is shown in Fig. 2. In this section, we first introduce the channel attention residual network for encoding and decoding. Then, we present a novel quantizer based on GMM. Finally, we illustrate the details of the variable quantization level controller, which makes the entire system able to dynamically alter the quantization levels for each channel.

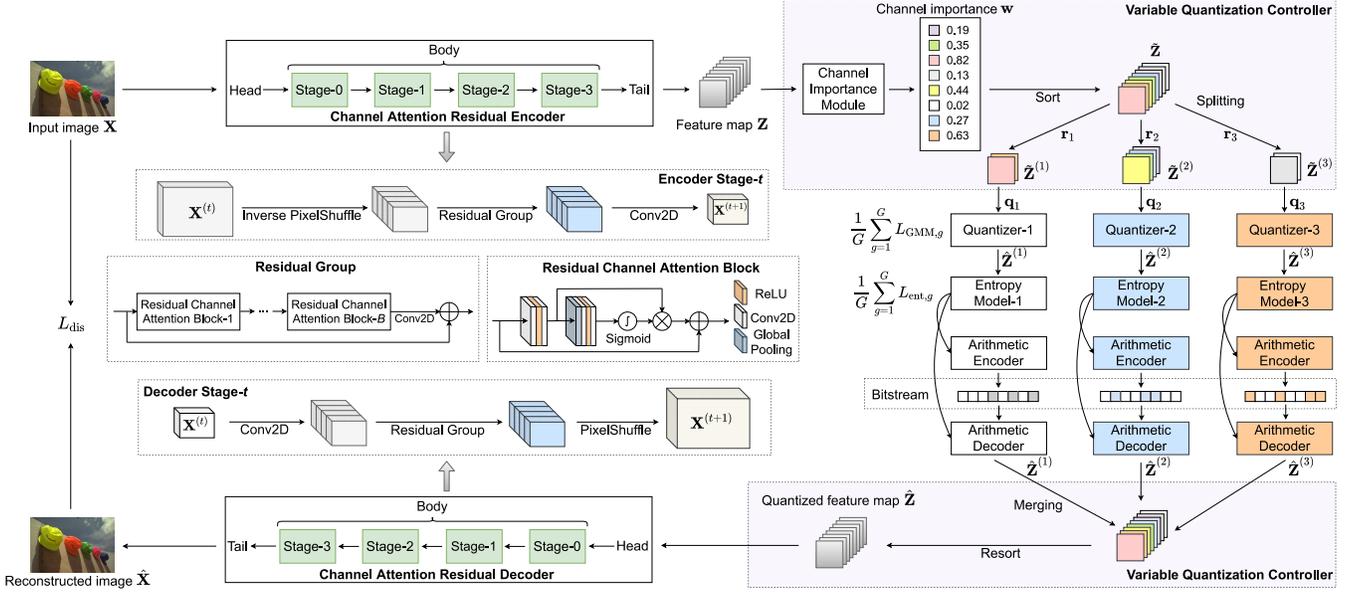


Figure 2: Framework of the channel-level variable quantization network. The entire encoder and decoder bodies both contain four stages.  $C = 8$ ,  $G = 3$ , and  $\mathbf{r} = [25\%, 50\%, 25\%]^T$  are selected for the illustration of variable quantization controller. *Best viewed on screen.*

### 3.1 Channel Attention Residual En/Decoder

Our channel attention residual encoder comprises three parts: head, body, and tail. The head module contains one convolutional layer, which transforms the original image into feature map  $\mathbf{X}^{(0)}$  with  $C_0$  channels. The body of the encoder is shown in left part of Fig. 2. The entire body includes four stages. In each stage, the output feature map is only half the resolution  $(h, w)$  of the input feature map. We denote the input feature map at Stage- $t$  as  $\mathbf{X}^{(t)} \in \mathbb{R}^{C_t \times H \times W}$ . Motivated by the super-resolution task’s method [Shi *et al.*, 2016], we use inverse PixelShuffle to implement the down-sampling operation. It can be expressed as:

$$\mathcal{IPS}(\mathbf{X}^{(t)})_{c(di+j),h,w} = \mathbf{X}^{(t)}_{c,dh+i,dw+j}, 1 \leq i, j \leq d, \quad (1)$$

where  $d$  is the down-sampling factor. It is a periodic shuffling operator that rearranges the elements of a  $C_t \times H \times W$  tensor to a tensor of shape  $d^2 C_t \times \frac{1}{d} H \times \frac{1}{d} W$ . Notably, this operator preserves all information of the input because the number of elements does not vary. We also found that inverse PixelShuffle can improve the stability of training and reduce the memory costs relative to the down-sampling convolution.

Previous CNN-based image compression methods treat channel-wise features equally, which is not flexible for real cases. To make the network focus on more informative features, we follow [Hu *et al.*, 2018; Zhang *et al.*, 2018; Zhong *et al.*, 2018] and exploit the inter-dependencies among feature channels. We send the feature map to the residual group module, shown in left part of Fig. 2. The residual group consists of  $B$  residual channel attention blocks, which are used to extract the inter-dependencies among feature channels and distill the feature map. The residual group does not change the number of channels.

Finally, we add a convolutional layer to alter the number of channels from  $C_t$  to  $C_{t+1}$  for the next stage. Thus the output

of Stage- $t$  is  $\mathbf{X}^{(t+1)} \in \mathbb{R}^{C_{t+1} \times \frac{1}{d} H \times \frac{1}{d} W}$ , which is also the input of the next stage.

After four stages of processing in the body, a convolutional layer, appended as the tail part, generates the compressed (latent) representation  $\mathbf{Z}$  with  $C$  channels, where  $C$  can be varied manually for different BPPs. Similarly, the architecture of the decoder is simply the inverse version of the encoder. As shown in left part of Fig. 2, we replace inverse PixelShuffle with PixelShuffle for the up-sampling operation.

### 3.2 GMM Quantizer

For the quantizer, we propose a novel quantization method based on GMM. Concretely, we model the prior distribution  $p(\mathbf{Z})$  as a mixture of Gaussian distributions:

$$p(\mathbf{Z}) = \prod_i \sum_{q=1}^Q \pi_q \mathcal{N}(z_i | \mu_q, \sigma_q^2), \quad (2)$$

where  $\pi_q$ ,  $\mu_q$ , and  $\sigma_q$  are the learnable mixture parameters and  $Q$  is the quantization level.

We obtain the forward quantization result by setting it to the mean that takes the largest responsibility:

$$\hat{z}_i \leftarrow \arg \max_{\mu_j} \frac{\pi_j \mathcal{N}(z_i | \mu_j, \sigma_j^2)}{\sum_q \pi_q \mathcal{N}(z_i | \mu_q, \sigma_q^2)}. \quad (3)$$

Obviously, Eqn. (3) is non-differentiable. We relax  $\hat{z}_i$  to  $\tilde{z}_i$  to compute its gradients during the backward pass by:

$$\tilde{z}_i = \sum_{j=1}^Q \frac{\pi_j \mathcal{N}(z_i | \mu_j, \sigma_j^2)}{\sum_q \pi_q \mathcal{N}(z_i | \mu_q, \sigma_q^2)} \mu_j. \quad (4)$$

Unlike the conventional GMM, which optimizes  $\pi_q$ ,  $\mu_q$ , and  $\sigma_q$  by using the expectation maximization (EM) algorithm, we learn the mixture parameters by minimizing the

negative likelihood loss function through the network back-propagation. We denote the prior distribution loss function of GMM quantizer as:

$$L_{\text{GMM}} = -\log(p(\mathbf{Z})) = -\sum_i \log \sum_{q=1}^Q \pi_q \mathcal{N}(z_i | \mu_q, \sigma_q^2). \quad (5)$$

Here, we would like to make a comparison between the GMM quantizer and the soft quantizer [Agustsson *et al.*, 2017]. The soft quantizer can be viewed as a differentiable version of the K-means algorithm. If the mixture parameters satisfy:  $\pi_1 = \pi_2 = \dots = \pi_Q = 1/Q$  and  $\sigma_1 = \sigma_2 = \dots = \sigma_Q = \sqrt{2}/2$ , the GMM quantizer will degenerate to the soft quantizer, which implies that the GMM quantizer has a more powerful representation and is a more generalized model.

### 3.3 Variable Quantization Controller

As mentioned in Sec. 2, each channel of the quantized feature map may have a different impact on the final reconstruction results. To allocate appropriate bitrates for different channels, we propose the variable quantization controller model.

The illustration of the variable quantization controller is shown in the right part of Fig. 2. In the variable quantization controller, there are two key components: channel importance module and splitting-merging module. In the following, we will introduce the mechanism of these two modules in detail.

#### Channel Importance Module

The input of the channel importance module is  $\mathbf{Z}$ , which is the output of the encoder mentioned in Sec. 3.1. Let us denote the channel number of  $\mathbf{Z}$  as  $C$  ( $C = 8$  in Fig. 2). We expect the channel importance module to generate a channel importance vector  $\mathbf{w} \in \mathbb{R}_+^C$ . Each element  $w_c$  represents the reconstruction importance of Channel- $c$ .

Here, we design three types of channel importance module:

**Sequeeze and excitation block-based.** We utilize average pooling and two convolutional layers to operate  $\mathbf{Z}$  (refer [Hu *et al.*, 2018]) and get an  $M \times C$  matrix, where  $M$  is the mini-batch size. We generate a learnable channel importance vector  $\mathbf{w}$  by using the mean operation on the matrix by reducing the first dimension ( $M$ ).

**Reconstruction error-based.** We perform three steps to implement it: First, we construct a validation dataset by randomly selecting  $N$  images from the training dataset. Second, we prune the  $c$ -th channel of the  $n$ -th image's feature map  $\mathbf{Z}_{n,c}$ :  $\mathbf{Z}_{n,c}(c, :, :) = 0$ . Last, we represent  $w_c$  by calculating the average MS-SSIM reconstruction error of each channel over the validation dataset:

$$w_c = \frac{1}{N} \sum_{n=1}^N d_{\text{MS-SSIM}}(\mathbf{I}_n, \text{Dec}(\text{Qua}(\mathbf{Z}_{n,c}))), \quad (6)$$

where  $\mathbf{I}_n$  is the  $n$ -th image of the validation dataset,  $\text{Dec}$  and  $\text{Qua}$  are represent the decoder and quantizer, respectively.

**Predefined.** We directly predefine the channel importance vector  $\mathbf{w}$  as  $w_c = c$ , which is fixed during the training and evaluation process.

#### Splitting-Merging Module

At the beginning of the splitting-merging module, we sort the feature map  $\mathbf{Z}$  in ascending order according to the channel importance vector  $\mathbf{w}$ . Because the new feature map is well organized, we split it to  $G$  groups ( $G = 3$  in Fig. 2). The  $G$  portions of the feature map are quantized and encoded using different quantization levels in different groups.

After the splitting operation, the  $C$  channels are divided into  $G$  groups. We denote the ratio vector of  $G$  groups as  $\mathbf{r}$ , which satisfies:  $\sum_g \mathbf{r}_g = 1$ , and  $\forall g, \mathbf{r}_g > 0$ . Here, we use the right part of Fig. 2 to explain its mechanism. Suppose that the parameters  $C = 8$ ,  $G = 3$ , and  $\mathbf{r} = [25\%, 50\%, 25\%]^\top$ , Channel-1 and 2 will be assigned Group-1 for quantization and encoding, Channel-3, 4, 5, and 6 will be assigned Group-2 and Channel-7 and 8 will be assigned Group-3. On the other hand, because the channel importances of Channel-1 and 2 are smaller than the others, we use smaller quantization level  $\mathbf{q}_1$  for quantizing and encoding. Similarly, we apply a larger quantization level  $\mathbf{q}_3$  to quantize and encode Channel-7 and 8. At the last step, we merge  $G$  groups and reorder the channel dimension to construct the final compressed result.

#### Analysis

Here, we conduct an analysis, examining under what condition the variable quantization controller can theoretically guarantee a better compression rate than that of the original one-group model. We suppose that the feature map  $\mathbf{Z}$  is a  $C \times H \times W$  tensor and the channel number of Group- $g$  is  $C_g = C\mathbf{r}_g$ . Obviously, it satisfies  $\sum_g C_g = C$ .  $\hat{\mathbf{Z}}$  has the same dimensions as  $\mathbf{Z}$  because  $\hat{\mathbf{Z}}$  is simply the quantized version of  $\mathbf{Z}$ . Because the number of dimensions  $\dim(\hat{\mathbf{Z}})$  and the quantization level  $Q$  are finite, the entropy is bounded by  $H(\hat{\mathbf{Z}}) \leq \dim(\hat{\mathbf{Z}})\log_2(Q) = CHW\log_2(Q)$  (refer, e.g., [Cover and Thomas, 2012]). Contrastingly, for  $G$  groups, suppose that the quantization level vector is  $\mathbf{q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_G]^\top$ , then, the entropy upper-bound of  $\{\hat{\mathbf{Z}}_g\}$  is:

$$H(\{\hat{\mathbf{Z}}_g\}) = \sum_{g=1}^G H(\hat{\mathbf{Z}}_g) \leq HWC \sum_{g=1}^G \mathbf{r}_g \log_2(\mathbf{q}_g). \quad (7)$$

Thus, if the  $G$  groups satisfy  $\mathbf{r}^\top \log_2(\mathbf{q}) < \log_2(Q)$ , the variable quantization controller will provide a lower entropy upper-bound than the conventional one-group model. On the other hand, although  $\hat{\mathbf{Z}}$  has the same total number of elements as  $\{\hat{\mathbf{Z}}_g\}$ ,  $\hat{\mathbf{Z}}$  has only  $Q$  values to pick up, whereas  $\{\hat{\mathbf{Z}}_g\}$  has  $\sum_g \mathbf{q}_g$  values, indicating that  $\{\hat{\mathbf{Z}}_g\}$  may have better diversity.

Overall, in the variable quantization controller, we choose the GMM quantizer (in Sec. 3.2) and the 3D CNN-based context model (refer [Mentzer *et al.*, 2018]) for quantization, and entropy estimating, respectively. All quantized feature maps  $\{\hat{\mathbf{Z}}_k\}$  will concatenate together and be sent to the decoder. The final loss function of the entire system becomes:

$$L = \alpha L_{\text{dis}} + \frac{1}{G} \sum_{g=1}^G L_{\text{ent},g} + \beta \frac{1}{G} \sum_{g=1}^G L_{\text{GMM},g}. \quad (8)$$

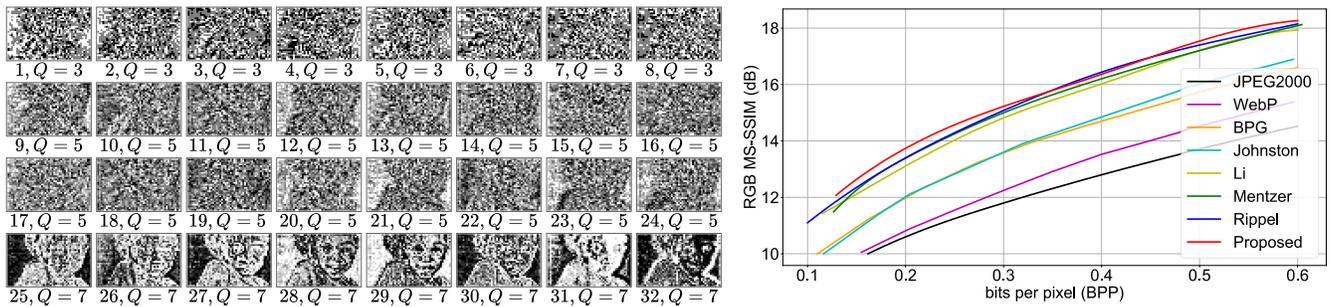


Figure 3: Left: Visualization results (*kodim15*) of the predefined model’s quantized feature map, which contains three quantization levels: 3, 5, and 7. Right: Comparisons of the rate-distortion curves on Kodak. MS-SSIM values are converted into decibels. *Best viewed on screen.*

$\mathbf{q}$	CI Type	MS-SSIM	BPP
[5]		0.9651	0.2664
[3, 5, 7] <sup>T</sup>	SE-based	0.9646	0.2608 (↓ 2.11%)
[3, 5, 7] <sup>T</sup>	RE-based	0.9652	0.2586 (↓ 2.93%)
[3, 5, 7] <sup>T</sup>	Predefine	<b>0.9653</b>	<b>0.2576</b> (↓ <b>3.31%</b> )

Table 1: Investigation of channel importance module. We run it three times and show the average results. CI Type denotes the type of channel importance module mentioned in Sec.3.3.

## 4 Experiments

### 4.1 Implementation and Training Details

**Datasets.** We merge three common datasets, namely DIK2K [Timofte *et al.*, 2017], Flickr2K [Lim *et al.*, 2017], and CLIC2018, to form our training dataset, which contains approximately 4,000 images in total. Following many deep image compression methods, we evaluate our models on the Kodak dataset with the metrics MS-SSIM for lossy image compression.

**Parameter setting.** In our experiments, we use the Adam optimizer [Kingma and Ba, 2015] with a mini-batch size  $M$  of 32 to train our five models on  $256 \times 256$  image patches. We vary the quantized feature map  $\hat{\mathbf{Z}}$ ’s channel number  $C$  from 16 to 80 to obtain different BPPs. The total number of training epochs equals to 400. The initialized learning rates are set to  $1 \times 10^{-4}$ ,  $1 \times 10^{-4}$ ,  $5 \times 10^{-5}$  and  $1 \times 10^{-4}$  for the encoder, quantizer, entropy model, and decoder, respectively. We reduce them twice (at Epoch-200 and Epoch-300) by a factor of five during training. In the channel attention residual en/decoder, we set the number of residual channel attention blocks  $B = 6$  for all stages. The channel numbers for each stage in the encoder are 32, 64, 128, and 192, respectively, whereas those for each stage in the decoder are 192, 128, 64, and 32, respectively. In the variable quantization controller, we set the number of groups  $G = 3$ . The ratio vector  $\mathbf{r} = [25\%, 50\%, 25\%]^T$ . For loss function Eqn. (8), we choose negative MS-SSIM for the distortion loss  $L_{\text{dis}}$  and  $\alpha = 128$ ; we select cross entropy for the entropy estimation loss  $L_{\text{ent}}$  and  $\beta = 0.001$ .

### 4.2 Ablation Study

#### Investigation of Channel Importance Module

To demonstrate the effectiveness of the variable quantization mechanism and the channel importance module, we design

$\mathbf{q}$	PSNR	MS-SSIM	BPP
[5]	27.926	0.9651	0.2664
[4, 5, 6] <sup>T</sup>	28.012	0.9652	0.2639(↓ 0.94%)
[3, 5, 7] <sup>T</sup>	<b>28.024</b>	<b>0.9653</b>	0.2576(↓ 3.31%)
[2, 5, 8] <sup>T</sup>	27.982	0.9644	<b>0.2471</b> (↓ <b>7.24%</b> )

Table 2: Investigation of the combination in  $\mathbf{q}$ . We run it three times and show the average results.

several comparative experiments to evaluate the reconstruction performance. The baseline model generated a quantized feature map with channel number  $C = 32$ . The quantization level vector  $\mathbf{q} = [5]$  indicates that there are no splitting and merging operations. Thus, this model just contains one group. By contrast, with the same setting  $\mathbf{q} = [3, 5, 7]^T$  and  $\mathbf{r}$ , we use three different types of the channel importance module mentioned in Sec. 3.3, i.e., Squeeze and excitation block (SE)-based, reconstruction error (RE)-based, and predefined. We train these four variants for 400 epochs under the same training setting. We run all experiments three times and record the best MS-SSIM on Kodak. The details of the average results are listed in Tab. 1. We observe that the channel importance module and the splitting-merging module make the system more effective (smaller BPP) and powerful (better MS-SSIM). Additionally, the predefined channel importance module distinctly outperforms SE and RE-based modules, even SE and RE-based modules are learnable and data-dependent. This may be consistent with the network pruning research [Liu *et al.*, 2019]: training predefined target models from scratch can have better performance than pruning algorithms under some conditions. We also visualize the quantized feature map of the predefined model in Fig. 3. Comparing it with Fig. 1 (top right), we can see that the channels containing much more profile and context information of the original image are allocated more bits in the new system.

#### Investigation of the Combination in $\mathbf{q}$

As mentioned in Sec. 3.3, if the  $G$  groups satisfy  $\mathbf{r}^T \log_2(\mathbf{q}) < \log_2(Q)$ , the variable quantization controller will provide a lower theoretical entropy upper-bound. Here, we explore what combination may have better performance. The baseline model only has one quantization level, i.e.,  $\mathbf{q} = [5]$ . We extend it to three types of combinations:  $\mathbf{q} = [4, 5, 6]^T$ ,  $\mathbf{q} = [3, 5, 7]^T$ , and  $\mathbf{q} = [2, 5, 8]^T$ . The ratio vectors of the three types of models are the same and equal to  $[25\%, 50\%, 25\%]^T$ . Quantitatively,  $\log_2(2) + \log_2(8) < \log_2(3) + \log_2(7) < \log_2(4) + \log_2(6) < 2\log_2(5)$ , and the

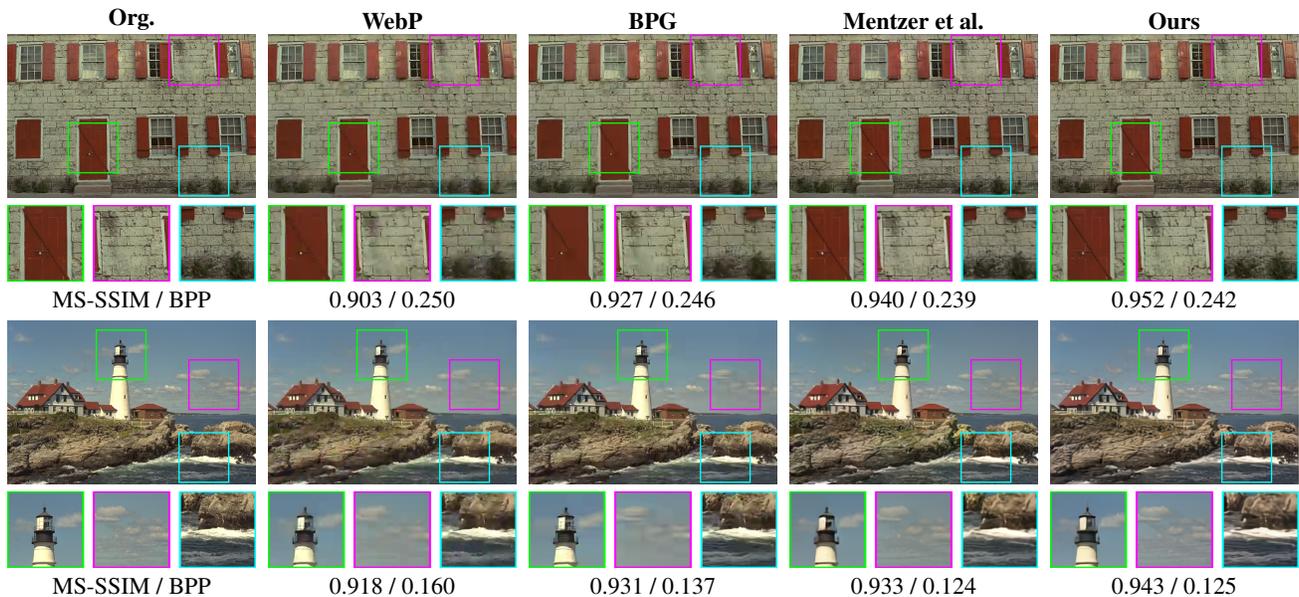


Figure 4: Visual comparisons on example images (top: *kodim1*, bottom: *kodim21*) from the Kodak dataset. From left to right: the original images, WebP, BPG, Mentzer’s, and ours. Our model achieves the best visual quality, demonstrating the superiority of our model in preserving both sharp edges and detailed textures. *Best viewed on screen.*

experimental results are consistent with the theoretical analysis. Additionally, we find that the odd quantized level may have better performances. Because the odd quantized level more likely contains a quantized value close to 0. This may meet the similar results in research related to network quantization [Zhu *et al.*, 2017]. If the quantization levels in  $\mathbf{q}$  are too different, e.g.,  $[2, 5, 8]^T$ , the performance will degrade.

### 4.3 Comparisons

In this subsection, we compare the proposed method against three conventional compression techniques, JPEG2000, WebP, and BPG (4:4:4), as well as recent deep learning-based compression work by [Johnston *et al.*, 2018], [Rippel and Bourdev, 2017], [Li *et al.*, 2018], and [Mentzer *et al.*, 2018]. We use the best performing configuration we can find of JPEG 2000, WebP, and BPG. Trading off between the distortion and the compression rate,  $\mathbf{q}$  is set to  $[3, 5, 7]^T$  in the following experiments.

#### Quantitative Evaluation

Following [Rippel and Bourdev, 2017; Mentzer *et al.*, 2018], and because MS-SSIM is more consistent with human visual perception than PSNR, we use MS-SSIM as the performance metric. Fig. 3 depicts the rate-distortion curves of these eight methods. Our method outperforms conventional compression techniques JPEG2000, WebP and BPG, as well as the deep learning-based approaches of [Toderici *et al.*, 2017], [Li *et al.*, 2018], [Mentzer *et al.*, 2018], and [Rippel and Bourdev, 2017]. This superiority of the proposed method holds for almost all tested BPPs, i.e., from 0.1 BPP to 0.6 BPP. It should be noted that both [Li *et al.*, 2018] and [Mentzer *et al.*, 2018] are trained on the Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) [Russakovsky *et al.*, 2015], which contains more than one million images. [Rippel and Bourdev, 2017] trained their models on the Yahoo Flickr Creative Commons 100 Million dataset [Thomee *et al.*, 2016],

which includes approximately 100 million images. While our models are trained using only 4,000 images.

#### Visual Quality Evaluation

Owing to the lack of reconstruction results for many deep image compression algorithms and the space limitations of the paper, we present only two reconstruction results of images and compare them with WebP, BPG, and [Mentzer *et al.*, 2018]. In the first row of Fig. 4, our method accurately reconstructs more clear and textural details of objects, e.g., door and the stripes on the wall. Other results have blocking artifacts more or less. For the second reconstruction results, our method can obtain better visual quality on images of objects such as clouds and water waves. Notably, our method is the only one that succeeds in reconstructing the spire of a lighthouse. Furthermore, the MS-SSIM measurements are also better than other methods in similar BBP ranges.

### 5 Conclusion

In this paper, we propose, to the best of our knowledge, the first channel-level method for deep image compression. Moreover, based on the channel importance module and the splitting-merging module, the entire system can variably allocate different bitrates to different channels, which can further improve the compression rates and performances. Additionally, we formulate the quantizer into a GMM manner, which is a universal pattern for the existing finite range quantizers. Ablation studies validate the effectiveness of the proposed modules. Extensive quantitative and qualitative experiments clearly demonstrate that our method achieves superior performance and generates better visual reconstructed results than the state-of-the-art methods without a hyperprior VAE.

#### Acknowledgments

This work was partially supported by JST CREST JPMJ-CR1686, Japan.

## References

- [Agustsson *et al.*, 2017] Eiríkur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, et al. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *NeurIPS*, pages 1141–1151, 2017.
- [Agustsson *et al.*, 2019] Eiríkur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. Generative adversarial networks for extreme learned image compression. In *ICCV*, pages 221–231, 2019.
- [Ballé *et al.*, 2017] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. In *ICLR*, 2017.
- [Ballé *et al.*, 2018] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *ICLR*, 2018.
- [Cover and Thomas, 2012] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.
- [Dong *et al.*, 2014] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, pages 184–199, 2014.
- [He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.
- [He *et al.*, 2017] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, pages 1389–1397, 2017.
- [Hu *et al.*, 2018] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141, 2018.
- [Johnston *et al.*, 2018] Nick Johnston, Damien Vincent, David Minnen, Michele Covell, Saurabh Singh, et al. Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. In *CVPR*, pages 4385–4393, 2018.
- [Kingma and Ba, 2015] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.
- [Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pages 1097–1105, 2012.
- [Lee *et al.*, 2019] Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context-adaptive entropy model for end-to-end optimized image compression. In *ICLR*, 2019.
- [Li *et al.*, 2018] Mu Li, Wangmeng Zuo, Shuhang Gu, Debin Zhao, and David Zhang. Learning convolutional networks for content-weighted image compression. In *CVPR*, pages 3214–3223, 2018.
- [Lim *et al.*, 2017] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, pages 136–144, 2017.
- [Liu *et al.*, 2017] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, pages 2736–2744, 2017.
- [Liu *et al.*, 2019] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *ICLR*, 2019.
- [Luo *et al.*, 2017] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. ThiNet: A filter level pruning method for deep neural network compression. In *ICCV*, pages 5058–5066, 2017.
- [Mentzer *et al.*, 2018] Fabian Mentzer, Eiríkur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *CVPR*, pages 4394–4402, 2018.
- [Minnen *et al.*, 2018] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *NeurIPS*, pages 10771–10780, 2018.
- [Rippel and Bourdev, 2017] Oren Rippel and Lubomir Bourdev. Real-time adaptive image compression. In *ICML*, pages 2922–2930, 2017.
- [Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.
- [Shi *et al.*, 2016] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, et al. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, pages 1874–1883, 2016.
- [Skodras *et al.*, 2001] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The JPEG 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58, 2001.
- [Thomee *et al.*, 2016] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.
- [Timofte *et al.*, 2017] Radu Timofte, Eiríkur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. NTIRE 2017 challenge on single image super-resolution: Methods and results. In *CVPR Workshops*, pages 114–125, 2017.
- [Toderici *et al.*, 2016] George Toderici, Sean M O’Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. Variable rate image compression with recurrent neural networks. In *ICLR*, 2016.
- [Toderici *et al.*, 2017] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. In *CVPR*, pages 5306–5314, 2017.
- [Van Oord *et al.*, 2016] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, pages 1747–1756, 2016.
- [Wallace, 1992] Gregory K Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1), 1992.
- [Wang *et al.*, 2003] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *Asilomar Conference on Signals, Systems & Computers*, volume 2, pages 1398–1402, 2003.
- [Zhang *et al.*, 2018] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, pages 286–301, 2018.
- [Zhong *et al.*, 2018] Zhisheng Zhong, Tiancheng Shen, Yibo Yang, Zhouchen Lin, and Chao Zhang. Joint sub-bands learning with clique structures for wavelet domain super-resolution. In *NeurIPS*, pages 165–175, 2018.
- [Zhu *et al.*, 2017] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. In *ICLR*, 2017.