DPICT: Deep Progressive Image Compression Using Trit-Planes

Jae-Han Lee[1,2] Seungmin Jeon[1] Kwang Pyo Choi[3] Youngo Park[3] Chang-Su Kim[1]
[1]Korea University [2]Gauss Labs [3]Samsung Electronics
{jaehanlee, seungminjeon}@mcl.korea.ac.kr,
{kp5.choi, youngo.park}@samsung.com, changsukim@korea.ac.kr

Abstract

We propose the deep progressive image compression using trit-planes (DPICT) algorithm, which is the first learning-based codec supporting fine granular scalability (FGS). First, we transform an image into a latent tensor using an analysis network. Then, we represent the latent tensor in ternary digits (trits) and encode it into a compressed bitstream trit-plane by trit-plane in the decreasing order of significance. Moreover, within each trit-plane, we sort the trits according to their rate-distortion priorities and transmit more important information first. Since the compression network is less optimized for the cases of using fewer trit-planes, we develop a postprocessing network for refining reconstructed images at low rates. Experimental results show that DPICT outperforms conventional progressive codecs significantly, while enabling FGS transmission. Codes are available at https://github.com/jaehanlee-mcl/DPICT.

1. Introduction

Image compression is a fundamental problem in image processing and analysis. Classical image codecs, such as JPEG [44], JPEG2000 [37], WebP [18], and BPG [8], have been developed to achieve the goal of efficient data storage and transmission. They contain several modules to process hand-crafted features. For example, for transform coding, JPEG uses discrete cosine transform, and JPEG2000 adopts wavelet transform.

Recently, with the availability of substantial training data and computing resources, deep learning has been adopted for image compression, as well as other image and vision problems. Some learning-based codecs are based on convolutional neural networks (CNNs) [5, 6, 26, 30], while others on recurrent neural networks (RNNs) [19, 24, 40, 41]. In terms of the rate-distortion (RD) performance, recent learning-based codecs [11, 13, 47] are competitive with or even superior to the classical codecs.

Progressive compression, or scalable coding [32], is a crucial issue. A progressive codec encodes an image into a single bitstream that can decoded at various bit-rates, as illustrated in Figure 1. There are various terminals from small wearable devices to big TVs, requiring different image qualities at different bit-rates. It is inefficient to encode multiple non-scalable bitstreams for these diverse devices. In contrast, a scalable bitstream can be efficiently truncated at multiple points to reconstruct the scene at different qualities. Moreover, when a network has a limited bandwidth, the receiver of a scalable bitstream can first check a preview image by receiving a small portion of the bitstream, and then reconstruct a higher quality image by decoding the remaining bits.

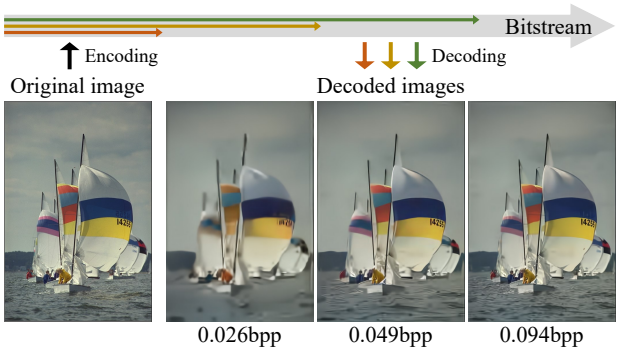


Figure 1. Illustration of progressive reconstruction of the proposed DPICT algorithm.

There are several learning-based progressive codecs [10, 19, 24, 40, 41], which, however, support coarse granular scalability only: a bitstream can be decoded at a limited number of rates. To the best of our knowledge, the proposed algorithm is the first learning-based codec with fine granular scalability (FGS) [28, 36]: a single bitstream can be truncated at any points to reconstruct the scene faithfully. Furthermore, despite this additional functionality, the proposed algorithm provides better RD performance than the conventional learning-based progressive codecs.

In this paper, we propose the deep progressive image codec using trit-planes (DPICT) with FGS. First, we transform an image into a latent tensor, each element of which is

represented by ternary digits (trits). Then, we encode the latent tensor trit-plane by trit-plane in the decreasing order of significance. Moreover, even in the same trit-plane, we sort the trits according to the RD priorities to transmit more important information first. At the decoder, when fewer trit-planes are used, the reconstructed image is degraded by quantization errors and contain noisy artifacts. To reduce such artifacts, we also develop postprocessing networks. Experimental results demonstrate that DPICT outperforms the conventional progressive codecs significantly, while supporting FGS.

2. Related Work

Learning-based compression: A typical learning-based image compression method [3,5,39] constructs a neural network, by integrating a quantizer into an autoencoder [43], which is trained end-to-end to minimize a loss function, including rate and distortion terms. Given an image, the encoder (or analysis network) generates a latent representation, which is then quantized. The decoder (or synthesis network) dequantizes the representation and reconstructs the image in a lossy manner. The quantizer is, however, not differentiable. For the backpropagation in the training phase, it is approximated by the binarization process [40], additive uniform noise [5], or stochastic rounding [39].

Rippel and Bourdev [35], Tschannen *et al*. [42], and Agustsson *et al*. [4] adopted generative adversarial networks [17] for image compression. Nakanishi *et al*. [31] developed an image codec based on a multi-scale autoencoder. Ballé *et al*. [6] proposed an additional autoencoder for a hyperprior. They assumed that latent elements have Gaussian distributions with zero mean, and used the hyperprior autoencoder to encode the standard deviations. Mentzer *et al*. [29], Minnen *et al*. [30], Lee *et al*. [26], and Li *et al*. [27] adopted context models. Also, more sophisticated hyperpriors have been studied. Cheng *et al*. [11] assumed Gaussian mixture models for latent elements, and Cui *et al*. [13] used asymmetric Gaussian distributions.

Variable-rate compression: While traditional codecs, such as JPEG [44], support variable bit-rates, most learning-based codecs [5,6,11,26,27,30,39] can generate bitstreams at fixed rates only. For multiple rates, they should train as many models, which incur inefficiency in testing, as well as in training.

Hence, several learning-based codecs have been developed to achieve variable-rate compression using a single network. Theis *et al*. [39] proposed a variable-rate training scheme for a single autoencoder using a scale parameter for quantization. Choi *et al*. [12] also employed the scaled quantization, while training rate-specific parameters for a few selected rates. Similarly, Yang *et al*. [48] proposed the modulated autoencoders containing separate modules for selected rates. Cai *et al*. [9] generated multi-scale representations and performed content-adaptive rate allocation. Cui *et al*. [13] proposed gain units to guide the network to allocate more bits to specific channels and also to control rates. Yang *et al*. [47] used the slimmable neural networks [49,50] to perform low-rate compression using only a fraction of the parameters and the highest-rate compression using all parameters. These codecs [9, 12, 13, 39, 47, 48] can adapt to different rates with a single trained model, but their bitstreams are not scalable, *i.e*. a lower-rate bitstream is not embedded in a higher-rate one.

Progressive compression: Progressive codecs to encode scalable bitstreams have also been studied. For example, JPEG [44] and JPEG2000 [37] are traditional progressive codecs. There are several learning-based progressive codecs, most of which are based on RNNs [19, 24, 40, 41]. Toderici *et al*. [40] proposed the first RNN-based progressive codec. Their network, utilizing the long short-term memory (LSTM) [20], transmits bits progressively: At stage $t+1$, the encoder transmits the residual error at stage t, and the decoder reconstructs it and adds it to the reconstructed image at stage t. If this is repeated T times, the single bitstream can support T different rates progressively. Gregor *et al*. [19] also introduced a recurrent image codec, which improves conceptual quality using a generative model. However, these codecs [19, 40] are for low-resolution patches. Torderici *et al*. [41] developed a codec for higher-resolution images by expanding the previous work in [40]. Johnston *et al*. [24] proposed an effective initializer for hidden states of their RNN and a spatially adaptive rate controller. However, all these RNN-based algorithms support only coarse granular scalability: a bitstream can be reconstructed at T different rates only, where T is the number of recurrent stages. Moreover, their compression performances are inferior even to those of the traditional codecs [8]. On the other hand, Cai *et al*. [10] proposed a network of a single encoder and two decoders. The encoder decomposes an image into two representations. Then, the preview decoder uses only one representation, and the full-quality decoder uses both representations. Hence, their network supports two rates only.

The proposed DPICT algorithm supports FGS in contrast to these learning-based progressive codecs [10, 19, 24, 40, 41]. Furthermore, DPICT provides significantly better RD performance than the conventional progressive codecs.

3. Proposed Algorithm

3.1. Compression network

We adopt the compression framework in Figure 2, which consists of an encoder g_a, a decoder g_s, a hyper encoder h_a, and a hyper decoder h_s, as done in [6, 11–13, 26, 30, 47]. An image $\mathbf{X}$ is transformed to a latent representation $\mathbf{Y}$ and

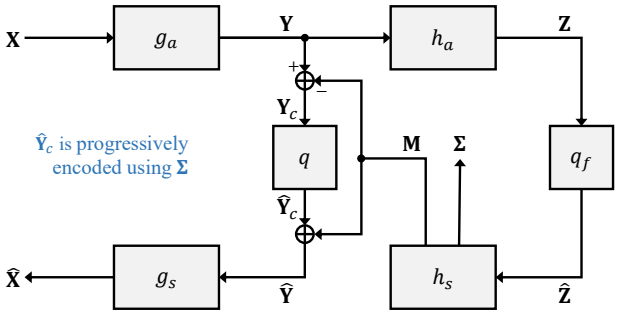


Figure 2. Image compression framework.

a hyper latent representation $\mathbf{Z}$ sequentially by g_a and h_a. Using the factorized prior model [6], denoted by $q_f(\cdot)$ in Figure 2, $\mathbf{Z}$ is digitized to $\hat{\mathbf{Z}}$. From $\hat{\mathbf{Z}}$, h_s yields $\mathbf{M}$ and $\boldsymbol{\Sigma}$, which contain the means and standard deviations of the elements in $\mathbf{Y}$, respectively. These elements are assumed to be independent Gaussian random variables. Then, the mean-removed (or centered) $\mathbf{Y}_c = \mathbf{Y} - \mathbf{M}$ is quantized to

$$\hat{\mathbf{Y}}_c = q(\mathbf{Y}_c) \tag{1}$$

where rounding is used for the quantizer $q(\cdot)$. Finally, the decoder g_s adds $\mathbf{M}$ back to $\hat{\mathbf{Y}}_c$ to yield

$$\hat{\mathbf{Y}} = \hat{\mathbf{Y}}_c + \mathbf{M} \tag{2}$$

and uses it to reconstruct $\hat{\mathbf{X}}$.

In addition to $\hat{\mathbf{Z}}$, $\hat{\mathbf{Y}}_c$ in (1) is encoded into a bitstream. Let $\hat{y}_c$, y_c, and σ be corresponding elements in $\hat{\mathbf{Y}}_c$, $\mathbf{Y}_c$, and $\boldsymbol{\Sigma}$. Then, the number of bits for encoding $\hat{y}_c$ is given by

$$N(\hat{y}_c) = -\log_2 P\left(\hat{y}_c - \tfrac{1}{2} \le y_c < \hat{y}_c + \tfrac{1}{2}\right) \tag{3}$$

where $y_c \sim \mathcal{N}(0, \sigma^2)$. Unlike conventional algorithms, we compress $\hat{\mathbf{Y}}_c$ progressively using trit-planes in Section 3.2.

In the training phase, since the quantizer is not differentiable, it is approximated by an additive noise function $u(\cdot)$,

$$\tilde{\mathbf{Y}} = u(\mathbf{Y}) = \mathbf{Y} + \mathcal{U}\left(-\tfrac{1}{2}, \tfrac{1}{2}\right) \tag{4}$$

where $\mathcal{U}(-\frac{1}{2}, \frac{1}{2})$ is a uniform noise tensor in range $(-\frac{1}{2}, \frac{1}{2})$. The noise function $u(\cdot)$ is assumed to generate white noise and not considered during the back-propagation of gradients [5]. Similarly, $\tilde{\mathbf{Z}}$ is obtained from $\mathbf{Z}$ [30], and then $\tilde{\mathbf{M}}$ and $\tilde{\boldsymbol{\Sigma}}$ are estimated. Finally, $\tilde{\mathbf{X}}$ is reconstructed from $\tilde{\mathbf{Y}}$.

The loss function ℓ consists of a distortion term ℓ_D and a rate term ℓ_R,

$$\ell = \ell_D(\mathbf{X}, \tilde{\mathbf{X}}) + \lambda \cdot \ell_R(\tilde{\mathbf{Y}}, \tilde{\mathbf{Z}}; \tilde{\mathbf{M}}, \tilde{\boldsymbol{\Sigma}}) \tag{5}$$

where ℓ_D is defined as the mean squared error between $\mathbf{X}$ and $\tilde{\mathbf{X}}$, and ℓ_R is the rate for encoding the elements in $\tilde{\mathbf{Y}}$ and $\tilde{\mathbf{Z}}$. Notice that the rate for $\tilde{\mathbf{Y}}$ is given by the sum of the numbers of bits $N(\tilde{y})$, similar to (3), but based on $\tilde{\mathbf{M}}$ and $\tilde{\boldsymbol{\Sigma}}$. In (5), λ is a Lagrangian multiplier to control the RD trade-off.

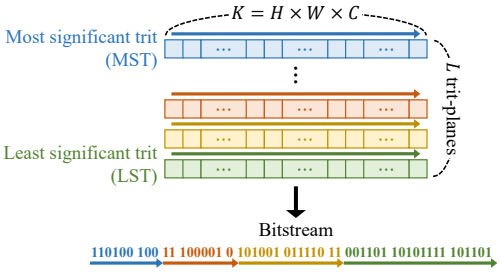


Figure 3. Illustration of the trit-plane coding of K elements.

3.2. Trit-plane coding

We represent $\hat{\mathbf{Y}}_c$ in (1) in a ternary number system and encode it trit-plane by trit-plane to achieve FGS. There are $K = H \times W \times C$ elements in $\hat{\mathbf{Y}}_c$, where H, W, and C are the height, the width, and the number of channels, respectively. Conventional algorithms transmit these K elements in a raster scan order, so they cannot decompress a partial bitstream meaningfully; the reconstructed image is severely degraded if some of the elements are missing. In contrast, we express each element using L trits and compress the L trit-planes in the decreasing order of significance from the most significant trit (MST) to the least significant trit (LST), as shown in Figure 3. In this way, front parts of the bitstream contain more important information, enabling the decoder to perform progressive reconstruction faithfully.

Figure 4 illustrates how to progressively compress an element $\hat{y}_c = q(y_c)$ in $\hat{\mathbf{Y}}_c$, where $y_c \sim \mathcal{N}(0, \sigma^2)$. In this example, $L = 3$, so it is assumed that y_c is rounded to one of the integers between -13 and $13 = \frac{3^L-1}{2}$. First, the number line is partitioned into three intervals, and the MST indicates which interval contains y_c. Specifically, $0_{(3)}$, $1_{(3)}$, and $2_{(3)}$ correspond to the left, middle, and right intervals, respectively. In Figure 4, y_c belongs to the middle interval $\mathcal{I}_1 = [-4.5, 4.5)$, so the trit $1_{(3)}$ is encoded into the bitstream. Second, $\mathcal{I}_1 = [-4.5, 4.5)$ is partitioned into three sub-intervals, and the next trit $2_{(3)}$ informs that $y_c \in \mathcal{I}_2 = [1.5, 4.5)$. Finally, $\mathcal{I}_2$ is partitioned again, and the third trit (LST) $0_{(3)}$ means that $y_c \in \mathcal{I}_3 = [1.5, 2.5)$. Hence, with all three trits, the decoder knows that $\hat{y}_c = 2$.

In general, let $\mathcal{I}_n = [l_n, r_n)$ denote the interval containing y_c when the first n trits are encoded. Also, let $t_n \in \{0_{(3)}, 1_{(3)}, 2_{(3)}\}$ denote the nth trit. $\mathcal{I}_n$ is partitioned into three sub-intervals of the same length (except for the leftmost and rightmost intervals):

$$\mathcal{I}_n^0 = [l_n^0, r_n^0) = [l_n, \tfrac{2l_n + r_n}{3}), \tag{6}$$

$$\mathcal{I}_n^1 = [l_n^1, r_n^1) = [\tfrac{2l_n + r_n}{3}, \tfrac{l_n + 2r_n}{3}), \tag{7}$$

$$\mathcal{I}_n^2 = [l_n^2, r_n^2) = [\tfrac{l_n + 2r_n}{3}, r_n). \tag{8}$$

Then, the next $(n+1)$th trit t_{n+1} reveals which subinterval contains y_c. It becomes $\mathcal{I}_{n+1}$.

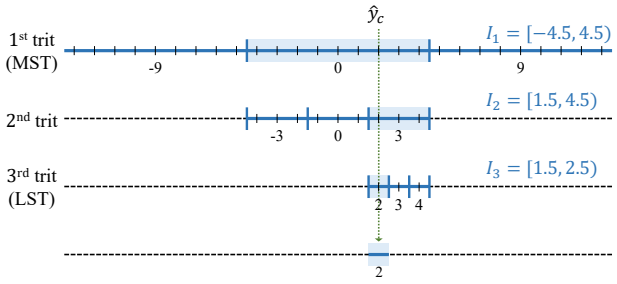


Figure 4. Trit-plane slicing of element $\hat{y}_c$.

The conditional probability of t_n given $\{t_{n-1}, \cdots, t_1\}$ is given by

$$P(t_n|t_{n-1}, \cdots, t_1) = P(y_c \in \mathcal{I}_n | y_c \in \mathcal{I}_{n-1}) \quad (9)$$

$$= \frac{\Phi(r_n/\sigma) - \Phi(l_n/\sigma)}{\Phi(r_{n-1}/\sigma) - \Phi(l_{n-1}/\sigma)} \quad (10)$$

where $\Phi(\cdot)$ is the CDF of the standard normal distribution. The number of bits for encoding t_n is then given by

$$N(t_n) = -\log_2 P(t_n|t_{n-1}, \cdots, t_1). \quad (11)$$

Also, by the chain rule,

$$P(\hat{y}_c) = P(t_L, \cdots, t_1) = \prod_{n=1}^{L} P(t_n|t_{n-1}, \cdots, t_1). \quad (12)$$

From (3), (11), and (12), we have

$$N(\hat{y}_c) = \sum_{n=1}^{L} N(t_n). \quad (13)$$

In other words, the trit-plane coding of $\hat{y}_c$ requires the same number of bits as the straightforward coding does. However, it allows progressive reconstruction of y_c. Suppose that the first n trits are received. Then, the decoder reconstructs y_c to

$$\hat{y}_c^n = E[y_c | y_c \in \mathcal{I}_n], \quad (14)$$

which is the minimum mean squared error (MMSE) estimate [16]. In other words, $\hat{y}_c^n$ in (14) yields the minimum mean squared distortion

$$D_n = E[(y_c - \hat{y}_c^n)^2 | y_c \in \mathcal{I}_n]. \quad (15)$$

Bit-plane coding: Bit-planes may be used instead of trit-planes. However, note that the most frequent $\hat{y}_c$ is 0 because $y_c \sim \mathcal{N}(0, \sigma^2)$. When $\hat{y}_c$ is 0, $\hat{y}_c^n$ in (14) is 0 regardless of n, which enables faithful reconstruction even at a low bitrate. This is impossible in the bit-plane coding, *i.e.* $\hat{y}_c^n \neq 0$ for $n \leq L-1$ when $\hat{y}_c = \hat{y}_c^L = 0$. Hence, we use trit-planes. It is shown in Section 4 that the trit-plane coding outperforms the bit-plane coding.

Coding of $\hat{\mathbf{Z}}$: We do not compress $\hat{\mathbf{Z}}$ progressively. Since the entropy coding of $\hat{y}$ depends on μ and σ estimated from $\hat{\mathbf{Z}}$, it is impossible to decompress $\hat{y}$ from a partially reconstructed $\hat{\mathbf{Z}}$. Also, $\hat{\mathbf{Z}}$ occupies only about 1% of the overall bitrate. Hence, we transmit all bits for $\hat{\mathbf{Z}}$ before the progressive transmission of $\hat{\mathbf{Y}}_c$.

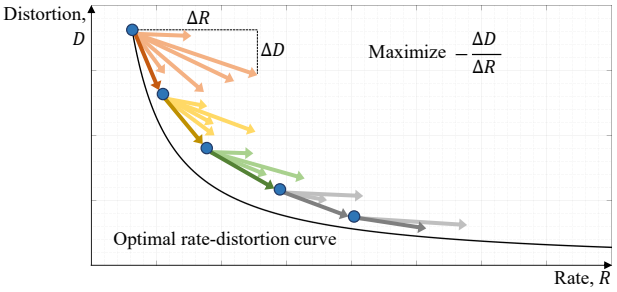


Figure 5. Greedy RD optimization using ratios $-\frac{\Delta D}{\Delta R}$.

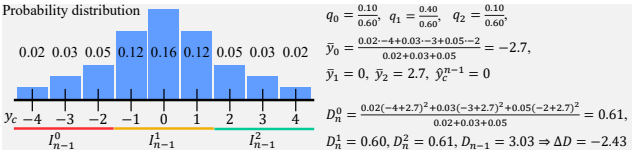


Figure 6. Toy example for computing $\bar{y}_k$, D_n^k, and ΔD.

3.3. RD-prioritized transmission

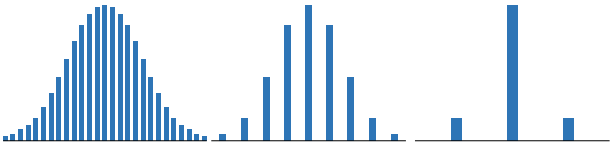
To transmit more important information first, we compress the L trit-planes from MST to LST. Moreover, within each trit-plane, we transmit the K trits after sorting them according to their RD priorities. The transmission of a trit increases the rate ($\Delta R > 0$), but it decreases the distortion ($\Delta D < 0$). The goal is to minimize ΔR and maximize $-\Delta D$ simultaneously. To achieve this goal, sophisticated RD methods [33] can be applied. However, for simplicity, we use the ratio $-\frac{\Delta D}{\Delta R}$ for RD-prioritized transmission.

In Figure 5, from the upper-left dot, there are several arrows corresponding to possible coding elements. We attempt to follow the optimal RD curve by selecting the element with the maximum ratio $-\frac{\Delta D}{\Delta R}$. After transmitting it, we repeat the process with the remaining elements. It is computationally prohibitive to measure ΔD exactly. Thus, we assume that the image distortion is proportional to the error in a latent element. The greedy selection and the distortion assumption cannot guarantee optimal transmission, but they yield good RD performance in practice, as will be shown in Section 4.

Suppose that the nth trit-plane is to be compressed. Thus, for an element $\hat{y}_c$, its first $n-1$ trits are already transmitted and its nth trit t_n is to be compressed. Both the encoder and the decoder can compute the probabilities $q_k = P(t_n = k|t_{n-1}, \cdots, t_1)$, $k = 0, 1, 2$ via (9). Then, the expected number of bits for encoding t_n is given by the entropy $H(\{q_0, q_1, q_2\})$,

$$\Delta R = H(\{q_0, q_1, q_2\}) = -\sum_{k=0}^{2} q_k \log_2 q_k. \quad (16)$$

We use the ANS coder [15] for the entropy coding. Also, for the three cases of $t_n = 0, 1, 2$, we compute the distortions via (15), respectively, which are denoted by D_n^0, D_n^1, D_n^2.



(a) $\hat{y}_c^L$ (b) $\hat{y}_c^{L-1}$ (c) $\hat{y}_c^{L-2}$

Figure 7. Histograms of (a) $\hat{y}_c^L$, (b) $\hat{y}_c^{L-1}$, and (c) $\hat{y}_c^{L-2}$.

(a) $\hat{\mathbf{X}}^L$ (b) $\hat{\mathbf{X}}^{L-3}$ (c) $\hat{\mathbf{X}}^{L-3} + \Delta\hat{\mathbf{X}}^{L-3}$

Figure 8. Reconstructed images (a) using all trit-planes and (b) using three fewer trit-planes. (c) Refined result of (b).

Then, the expected distortion change is given by

$$\Delta D = E[D_n] - D_{n-1} = \sum_{k=0}^{2} q_k D_n^k - D_{n-1} \tag{17}$$

where D_n^k is

$$D_n^k = E[(y_c - \bar{y}_k)^2 | y_c \in \mathcal{I}_{n-1}^k] \tag{18}$$

with $\bar{y}_k = E[y_c | y_c \in \mathcal{I}_{n-1}^k]$. Figure 6 illustrates how to compute $\bar{y}_k$, D_n^k, and ΔD with a toy example.

Finally, we compute the RD priority of the nth trit, which is defined as

$$-\frac{\Delta D}{\Delta R} = \frac{\sum_{k=0}^{2} q_k D_n^k - D_{n-1}}{\sum_{k=0}^{2} q_k \log_2 q_k}. \tag{19}$$

Then, we transmit the K trits in each trit-plane after sorting them in the decreasing order of their RD priorities. In this way, more important information is transmitted first, even in the same trit-plane.

3.4. Postprocessing network

To train the proposed algorithm, in (4), we use a uniform noise tensor in range $(-\frac{1}{2}, \frac{1}{2})$ to consider the case of using all L trit-planes. Thus, the network is less optimized for the cases of using fewer trit-planes. Figure 7 shows histograms of $\hat{y}_c^L$, $\hat{y}_c^{L-1}$, and $\hat{y}_c^{L-2}$. With fewer trit-planes, the gaps between reconstruction levels get wider, resulting in bigger quantization errors, which generate more artifacts in reconstructed images. Let $\hat{\mathbf{X}}^n$ denote a reconstructed image from a partial representation $\hat{\mathbf{Y}}_c^n$ consisting of $\hat{y}_c^n$'s. As shown in Figure 8(b), $\hat{\mathbf{X}}^{L-3}$ contains noisy artifacts.

We develop a postprocessing network g_p to reduce such artifacts. In other words, the goal of g_p is to convert a reconstructed image $\hat{\mathbf{X}}^n$ to a more faithful one $\hat{\mathbf{X}}^n + \Delta\hat{\mathbf{X}}^n$ with less artifacts. Figure 9 shows the training schema of g_p. In the upper pass, we first obtain $\hat{\mathbf{Y}}_c^n$ by (14) and then reconstruct $\hat{\mathbf{X}}^n = g_s(\hat{\mathbf{Y}}_c^n + \mathbf{M})$. Similarly, in the lower pass, we obtain the reconstructed image $\hat{\mathbf{X}}^L$ from $\hat{\mathbf{Y}}_c^L + \mathbf{M}$ using all trit-planes. We regard $\hat{\mathbf{X}}^L$ as a clean image without artifacts. Then, we train the postprocessing network to take $\hat{\mathbf{X}}^n$ and yield the residual $\Delta\hat{\mathbf{X}}^n$, by employing the postprocessing loss

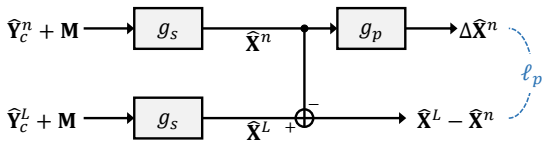


Figure 9. Training schema of a postprocessing network g_p.

$$\ell_p = \|\Delta\hat{\mathbf{X}}^n - (\hat{\mathbf{X}}^L - \hat{\mathbf{X}}^n)\|_2 \tag{20}$$

In the testing phase, using g_p, we refine a reconstructed image $\hat{\mathbf{X}}^n$ into $\hat{\mathbf{X}}^n + g_p(\hat{\mathbf{X}}^n)$. Figure 8(c) shows how g_p improves the reconstructed images $\hat{\mathbf{X}}^{L-3}$. We see that, compared to $\hat{\mathbf{X}}^{L-3}$, the refined images $\hat{\mathbf{X}}^{L-3} + \Delta\hat{\mathbf{X}}^{L-3}$ contain less artifacts and render the scenes more faithfully.

4. Experimental Results

4.1. Implementation

We develop a compression network similar to the Cheng *et al.*'s network [11], but we make some modifications to encode scalable bitstreams. First, we remove the autoregressive mask convolution, which predicts entropy parameters from a latent tensor. The autoregressive prediction assumes that the decoder can reconstruct the same latent tensor $\hat{\mathbf{Y}}$ in the raster scan order as the encoder does. However, in DPICT, the information in $\hat{\mathbf{Y}}$ is progressively transmitted. Therefore, before the decoder reconstructs all trit-planes, the assumption fails and the entropy decoding breaks down. Second, we assume that each latent element y has a Gaussian distribution, instead of a Gaussian mixture model. This is to simplify the MMSE estimation and RD optimization in (9) and (14)∼(19).

A postprocessing network g_p also has the encoder-decoder architecture. It is implemented with residual blocks, attention modules, and subpixel convolution layers [11]. However, g_p has 35 layers only, while the encoder g_a and the decoder g_s in the compression network in Figure 2 have 68 layers. We train two postprocessing networks, targeting at different bit-rates. Note that $\hat{\mathbf{X}}^n$ denotes a reconstructed image using the first n trit-planes. When n is not an integer, $\hat{\mathbf{X}}^n$ means that, in addition to the first $\lfloor n \rfloor$ trit-planes, $100 \times (n - \lfloor n \rfloor)$ % of the trits in the $(\lfloor n \rfloor + 1)$th trit-plane are used for the reconstruction. The first g_p targets at $\hat{\mathbf{X}}^n$ for $n \in [0, L-2.9]$, and the second g_p for $n \in (L-2.9, L-1.8]$.

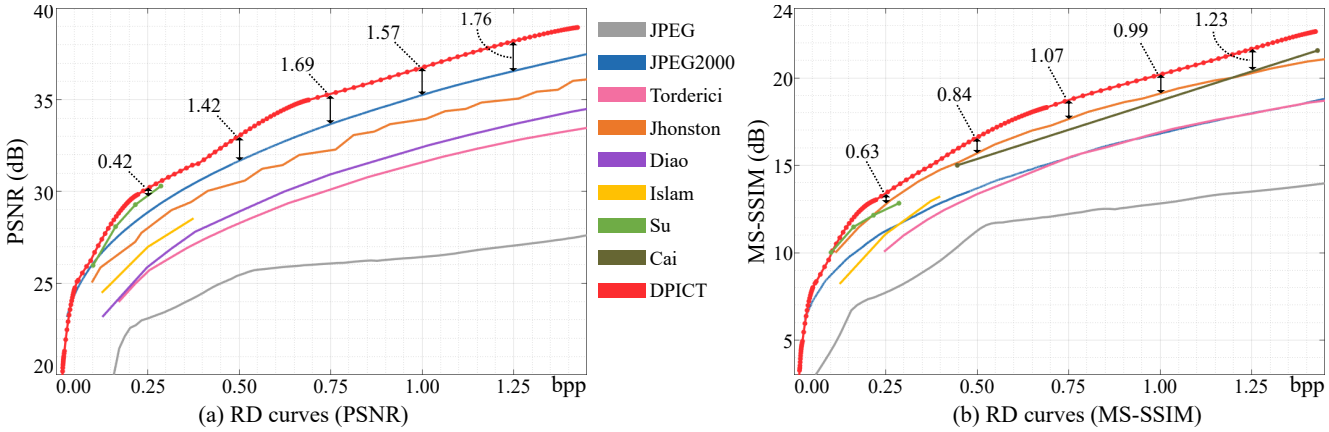


Figure 10. RD performance comparison of the proposed DPICT with conventional *progressive* codecs on the Kodak dataset: JPEG [44], JPEG2000 [37], Torderici *et al*. [41], Jhonston *et al*. [24], Diao *et al*. [14], Islam *et al*. [22], Su *et al*. [38], and Cai *et al*. [10]. At selected rates, the performance gaps between DPICT and the second best codecs are specified.

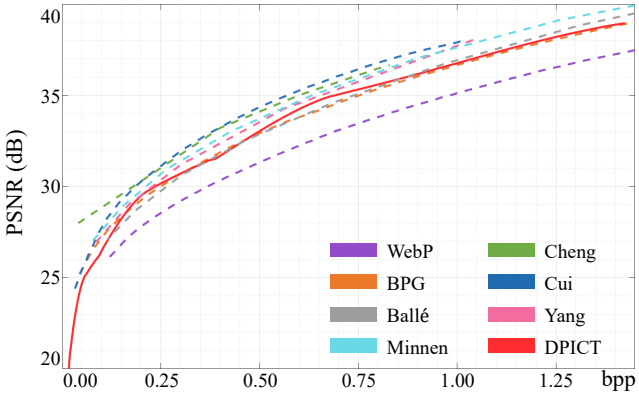


Figure 11. RD performance comparison of the proposed DPICT with *non-progressive* codecs on the Kodak dataset: WebP [18], BPG [8], Ballé *et al*. [6], Minnen *et al*. [30], Cheng *et al*. [11], Cui *et al*. [13], and Yang *et al*. [47].

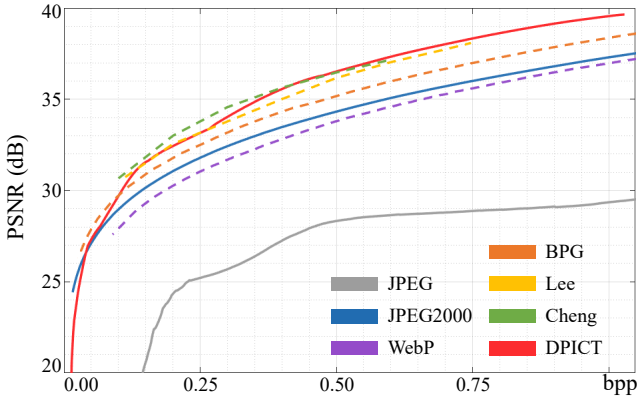


Figure 12. RD performance comparison on the CLIC dataset: DPICT is compared with JPEG [44], JPEG2000 [37], WebP [18], BPG [8], Lee *et al*. [26], and Cheng *et al*. [11]. Note that only JPEG, JPEG2000, and DPICT are progressive codecs, and their RD curves are solid lines.

We decided these ranges empirically and also observed that the postprocessing is not effective for $\hat{\mathbf{X}}^n$, $n \in (L-1.8, L]$, which is already of high quality.

For training, we use the Vimeo90k dataset [46]. Since it contains frames with overlapping contents, we sample 80,000 frames and randomly crop 256×256 patches from each sampled frame. We use the Adam optimizer [25] with a batch size of 16, a learning rate of $2 \cdot 10^{-5}$, and $\lambda = 5$. We perform the scheduled learning according to cosine annealing cycles [21]. We train the compression network for 200 epochs and then the postprocessing networks for 20 epochs.

For evaluation, we use the Kodak lossless image dataset [1] and the CLIC professional validation dataset [2]. The Kodak dataset consists of 24 images of resolution 512×768 or 768×512, while the CLIC dataset contains 41 images of higher quality up to 2K resolution. For each image, we determine the number L of trit-planes to cover the values of all elements, after truncating outliers, in $\hat{\mathbf{Y}}_c$. We measure the rate by bits per pixel (bpp), and the distortion by peak-to-signal ratio (PSNR) and multi-scale structural similarity (MS-SSIM) [45]. For MS-SSIM, we convert it to dB scale by MS-SSIM (dB) $= -10 \cdot \log_{10}(1 - \text{MS-SSIM})$.

We conduct all experiments using Pytorch [34] and CompressAI [7]. Network architecture and implementation details are available in the supplemental document.

4.2. Performance comparison

First, we compare the proposed DPICT with conventional progressive codecs: JPEG [44] and JPEG2000 [37] and learning-based progressive codecs [10,14,22,24,38,41]. Figure 10 plots the RD curves on the Kodak dataset. At every rate, DPICT provides the highest PSNR and the

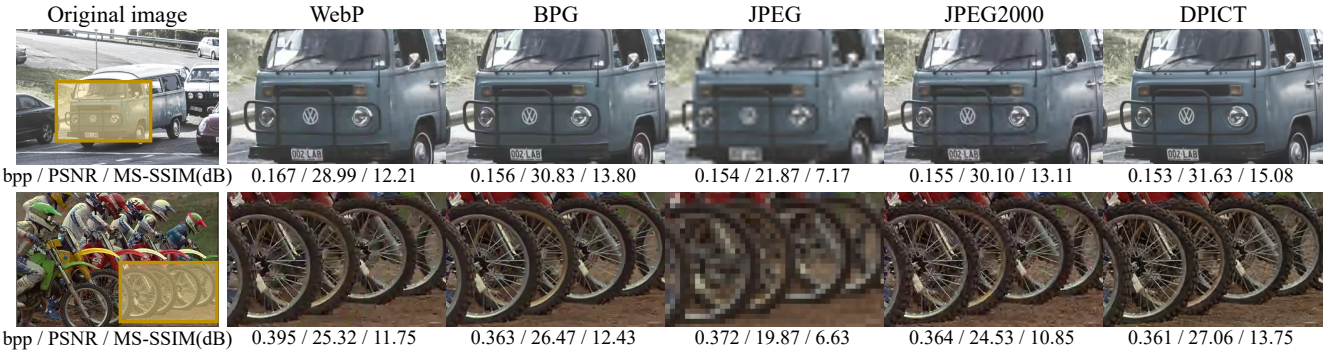


Figure 13. Qualitative comparison of reconstructed images at similar rates: WebP [18], BPG [8], JPEG [44], JPEG2000 [37], and DPICT.

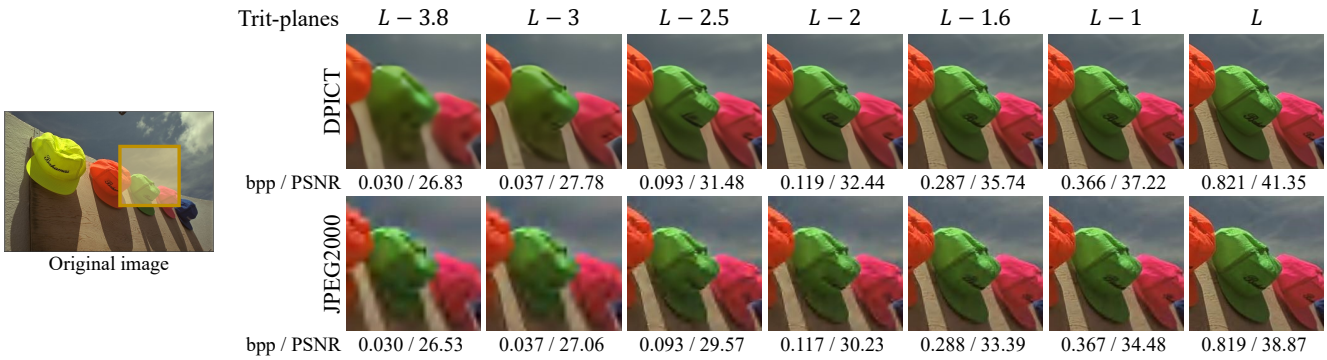


Figure 14. Qualitative comparison of progressively reconstructed images at various rates: JPEG2000 [37] and the proposed DPICT. At the top, the numbers of trit-planes used for the DPICT reconstruction are specified.

highest MS-SSIM, outperforming the second best codecs JPEG2000 and Jhonston *et al*. [24] considerably. For example, at 0.75bpp, DPICT yields about 1.7dB higher PSNR than JPEG2000, and about 1.1dB higher MS-SSIM than Jonston *et al*.

Moreover, it is worth pointing out that DPICT is the only learning-based codec with FGS. The same bitstream of DPICT for an image is reconstructed at 164 different rates, as indicated by red dots in Figure 10, which can be increased further if needed. On the other hand, Su *et al*. [38] support coarse granular scalability for four rates only, and their rate range is much narrower than that of DPICT. Cai *et al*. [10] support two rates only, one for preview images and the other for full-quality images. The RNN-based codecs [24, 41] support 16 rates, but their PSNR performances are poorer than those of JPEG2000.

Next, Figure 11 compares DPICT with non-progressive codecs [6, 8, 11, 13, 18, 30, 47]. Despite the additional functionality of FGS, DPICT outperforms the two traditional codecs, WebP and BPG, and is competitive with the state-of-the-art learning-based codecs.

Figure 12 compares RD curves on the CLIC dataset. We see that DPICT outperforms progressive codecs [37, 44] significantly and competes with Cheng *et al*. [11], which is a state-of-the-art non-progressive codec.

Figure 13 compares reconstructed images at similar rates. In areas with complicated texture and sharp edges, such as dirt floor or patterned window, the traditional codecs yield blurring artifacts, but DPICT restores high quality images without noticeable artifacts.

Figure 14 shows reconstructed images at different rates from a single bitstream. At the top of the figure, we specify how many trit-planes are decoded. At all rates, DPICT provides better qualities than the progressive JPEG2000.

4.3. Ablation study

We analyze the compression performance of DPICT in Figure 15. Here, each curve represents the result of replacing or removing certain components of DPICT.

First, we compare the proposed DPICT with four baselines excluding the trit-plane coding. In 'without sorting,' latent elements are transmitted channel by channel in the raster scan order without priority. This approach underperforms badly. In 'channel-wise sorting,' the C channels of the latent tensor are sorted and transmitted according to the RD priorities. Instead of channels, 'latent-wise sorting' sorts the $K = H \times W \times C$ latent elements. These alternatives can support progressive transmission, but are much inferior to DPICT. 'Bit-plane' is the result of replacing the trit-plane coding with the bit-plane coding. As mentioned in

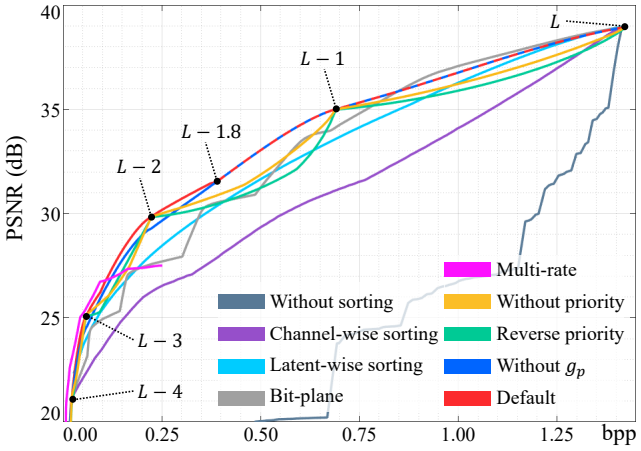


Figure 15. Ablation studies of DPICT on the Kodak dataset.

Section 3.2, $\hat{y}_c^n$ cannot be reconstructed to 0 in the bit-plane coding unless $n = L$. This causes significant performance degradation at low rates, *i.e.* when a bitstream is partially decoded. These results indicate that the trit-plane coding is an essential component of DPICT.

Second, we change the training schema. 'Multi-rate' is the result of changing the loss function so that the compression network is trained for multiple rates. Specifically, we replace the loss function in (5) with

$$\ell = \sum_{n=L-3}^{L} \left(\ell_D(\mathbf{X}, \tilde{\mathbf{X}}^n) + \lambda_n \ell_R(\tilde{\mathbf{Y}}^n, \tilde{\mathbf{Z}}; \tilde{\mathbf{M}}, \tilde{\mathbf{\Sigma}}) \right) \tag{21}$$

to consider partially reconstructed $\tilde{\mathbf{X}}^{L-3}$, $\tilde{\mathbf{X}}^{L-2}$, $\tilde{\mathbf{X}}^{L-1}$, as well as fully reconstructed $\tilde{\mathbf{X}}^L = \tilde{\mathbf{X}}$. Note that loss functions for different rates are also combined in [13, 47]. However, 'multi-rate' severely narrows the range of supported rates and is effective only at low rates.

Third, we replace the prioritized transmission. 'Without priority' is the result of not using the RD priority in (19). In this case, trits in each trit-plane are transmitted in the raster scan order. We see that the prioritized transmission is essential for reconstructing high quality $\hat{\mathbf{X}}^n$ when n is not an integer. 'Reverse priority' transmits the trits in a trit-plane in the increasing order of the RD priorities. It yields even poorer performance than 'without priority.'

Last, 'without g_p' is the result of not using the postprocessing networks. Note that the postprocessing networks improve the quality of a reconstructed image when the rate is lower than about 0.4bpp. Figure 16 shows the impacts of the postprocessing networks g_p. For easier comparison, improvement maps are also provided. When $L-2$ or fewer trit-planes are used, the postprocessing networks consistently improve reconstructed images both qualitatively and quantitatively. On the other hand, for reconstructed images using more than $L-2$ trit-planes, they do not provide clear improvements. Thus, we use the postprocessing networks

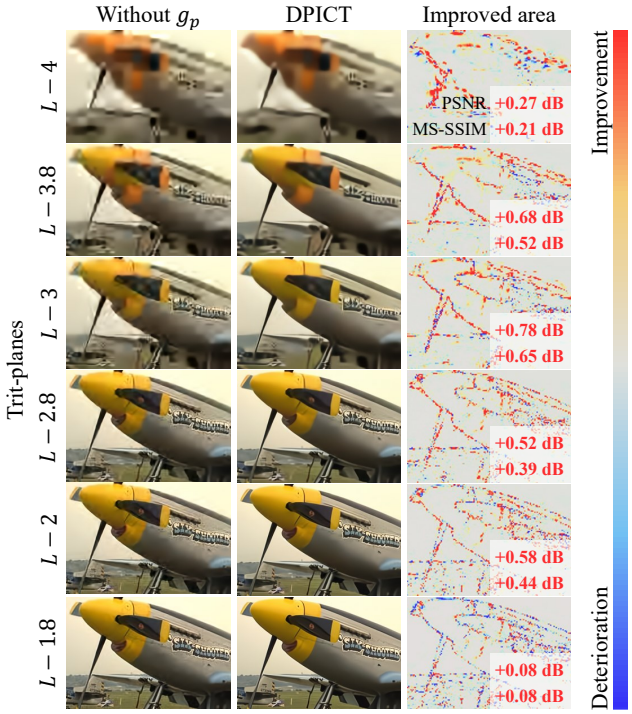


Figure 16. Comparison of reconstructed images before and after the postprocessing. Improvement maps are provided, where red and blue means improvement and deterioration, respectively.

when fewer than $L - 1.8$ trit-planes are decoded.

We provide more experimental results and analysis in the supplemental document. Also, we describe how to implement the proposed DPICT algorithm efficiently in [23].

5. Conclusions

In this paper, we proposed the DPICT algorithm supporting FGS. In DPICT, an image is transformed into a latent tensor using an analysis network. The latent tensor is then represented in a ternary number system and is encoded trit-plane by trit-plane in the decreasing order of significance. Furthermore, in each trit-plane, the trits are transmitted in the decreasing order of the RD priorities. We also developed the post-processing networks to reduce artifacts due to quantization errors when fewer trit-planes are used. Experiments demonstrated that the proposed DPICT provides state-of-the-art performance by outperforming other progressive codecs signficantly.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korea government (MSIT) (No. NRF-2021R1A4A1031864 and No. NRF-2022R1A2B5B03002310).

References

[1] *Kodak Lossless True Color Image Suite*. [Online]. Available: http://r0k.us/graphics/kodak. 6

[2] *Workshop and Challenge on Learned Image Compression*. 2018. http://challenge.compression.cc/tasks/. 6

[3] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc J. Van Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *NeurIPS*, 2017. 2

[4] Eirikur 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. 2

[5] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. In *ICLR*, 2017. 1, 2, 3

[6] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *ICLR*, 2018. 1, 2, 3, 6, 7

[7] Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. CompressAI: a PyTorch library and evaluation platform for end-to-end compression research. *arXiv preprint arXiv:2011.03029*, 2020. 6

[8] Fabrice Bellard. *BPG Image format*. 2014. https://bellard.org/bpg, Retrieved 2016-04-02. 1, 2, 6, 7

[9] Chunlei Cai, Li Chen, Xiaoyun Zhang, and Zhiyong Gao. Efficient variable rate image compression with multi-scale decomposition network. *IEEE Trans. Circuit Syst. Video Technol.*, 29(12):3687–3700, 2018. 2

[10] Chunlei Cai, Li Chen, Xiaoyun Zhang, Guo Lu, and Zhiyong Gao. A novel deep progressive image compression framework. In *IEEE PCS*, pages 1–5, 2019. 1, 2, 6, 7

[11] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *CVPR*, pages 7939–7948, 2020. 1, 2, 5, 6, 7

[12] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Variable rate deep image compression with a conditional autoencoder. In *ICCV*, pages 3146–3154, 2019. 2

[13] Ze Cui, Jing Wang, Shangyin Gao, Tiansheng Guo, Yihui Feng, and Bo Bai. Asymmetric gained deep image compression with continuous rate adaptation. In *CVPR*, pages 10532–10541, 2021. 1, 2, 6, 7, 8

[14] Enmao Diao, Jie Ding, and Vahid Tarokh. DRASIC: Distributed recurrent autoencoder for scalable image compression. In *IEEE DCC*, pages 3–12, 2020. 6

[15] Jarek Duda. Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding. *arXiv preprint arXiv:1311.2540*, 2013. 4

[16] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers Norwell, 1991. 4

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NeurIPS*, 27, 2014. 2

[18] Google. *WebP: Compression techniques*. 2017. https://developers.google.com/speed/webp/docs/compression. 1, 6, 7

[19] Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra. Towards conceptual compression. *NeurIPS*, 29:3549–3557, 2016. 1, 2

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997. 2

[21] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get m for free. In *ICLR*, 2017. 6

[22] Khawar Islam, L Minh Dang, Sujin Lee, and Hyeonjoon Moon. Image compression with recurrent neural network and generalized divisive normalization. In *CVPR Workshop*, pages 1875–1879, 2021. 6

[23] Seungmin Jeon, Jae-Han Lee, and Chang-Su Kim. RD-optimized trit-plane coding of deep compressed image latent tensors. *arXiv preprint arXiv:2203.13467*, 2022. 8

[24] Nick Johnston, Damien Vincent, David Minnen, Michele Covell, Saurabh Singh, Troy Chinen, Sung Jin Hwang, Joel Shor, and George Toderici. Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. In *CVPR*, pages 4385–4393, 2018. 1, 2, 6, 7

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[26] Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context-adaptive entropy model for end-to-end optimized image compression. In *ICLR*, 2019. 1, 2, 6

[27] Mu Li, Kede Ma, Jane You, David Zhang, and Wangmeng Zuo. Efficient and effective context-based convolutional entropy modeling for image compression. *IEEE Trans. Image Process.*, 29:5900–5911, 2020. 2

[28] Weiping Li. Overview of fine granularity scalability in MPEG-4 video standard. *IEEE Trans. Circuit Syst. Video Technol.*, 11(3):301–317, 2001. 1

[29] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *CVPR*, pages 4394–4402, 2018. 2

[30] David Minnen, Johannes Ballé, and George Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *NeurIPS*, 2018. 1, 2, 3, 6, 7

[31] Ken M. Nakanishi, Shin-ichi Maeda, Takeru Miyato, and Daisuke Okanohara. Neural multi-scale image compression. In *ACCV*, pages 718–732. Springer, 2018. 2

[32] J.-R. Ohm. Advances in scalable video coding. *Proc. IEEE*, 93(1):42–56, 2005. 1

[33] Antonio Ortega and Kannan Ramchandran. Rate-distortion methods for image and video compression. *IEEE Signal Process. Mag.*, 15(6):23–50, 1998. 4

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2018. 6

[35] Oren Rippel and Lubomir D. Bourdev. Real-time adaptive image compression. In *ICML*, 2017. 2

[36] Amir Said and William A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. Circuit Syst. Video Technol.*, 6(3):243–250, 1996. 1

[37] A. Skodras, C. Christopoulos, and T. Ebrahimi. The JPEG 2000 still image compression standard. *IEEE Signal Process. Mag.*, 18(5):36–58, 2001. 1, 2, 6, 7

[38] Rige Su, Zhengxue Cheng, Heming Sun, and Jiro Katto. Scalable learned image compression with a recurrent neural networks-based hyperprior. In *ICIP*, pages 3369–3373. IEEE, 2020. 6, 7

[39] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. In *ICLR*, 2017. 2

[40] 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. 1, 2

[41] 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. 1, 2, 6, 7

[42] Michael Tschannen, Eirikur Agustsson, and Mario Lucic. Deep generative models for distribution-preserving lossy compression. In *NeurIPS*, pages 5933–5944, 2018. 2

[43] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103, 2008. 2

[44] G. K. Wallace. The JPEG still picture compression standard. *IEEE Trans. Consum. Electron.*, 38(1):xviii–xxxiv, 1992. 1, 2, 6, 7

[45] Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402, 2003. 6

[46] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T. Freeman. Video enhancement with task-oriented flow. *Int. J. Comput. Vis.*, 127(8):1106–1125, 2019. 6

[47] Fei Yang, Luis Herranz, Yongmei Cheng, and Mikhail G. Mozerov. Slimmable compressive autoencoders for practical neural image compression. In *CVPR*, pages 4998–5007, 2021. 1, 2, 6, 7, 8

[48] Fei Yang, Luis Herranz, Joost Van De Weijer, José A. Iglesias Guitián, Antonio M. López, and Mikhail G. Mozerov. Variable rate deep image compression with modulated autoencoder. *IEEE Signal Process. Lett.*, 27:331–335, 2020. 2

[49] Jiahui Yu and Thomas S. Huang. Universally slimmable networks and improved training techniques. In *ICCV*, pages 1803–1811, 2019. 2

[50] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *ICLR*, 2018. 2