A. Softmax and Entropy

Theorem 1. *$H(\{p_0, p_1, p_2\})$ is a monotonic decreasing function of β, where*

$$p_i = \frac{e^{\beta y_i}}{\sum_{j=0}^{2} e^{\beta y_j}}, \qquad i = 0, 1, 2, \tag{8}$$

and $\beta > 0$.

Proof. Let $x = e^{\beta}$ and $A = x^{y_0} + x^{y_1} + x^{y_2}$. Then,

$$-H(\{p_0, p_1, p_2\}) = -H\left(\left\{\frac{x^{y_0}}{A}, \frac{x^{y_1}}{A}, \frac{x^{y_2}}{A}\right\}\right) \tag{9}$$

$$= \frac{x^{y_0}}{A}\log\frac{x^{y_0}}{A} + \frac{x^{y_1}}{A}\log\frac{x^{y_1}}{A} + \frac{x^{y_2}}{A}\log\frac{x^{y_2}}{A}. \tag{10}$$

The derivative of $-H$ with respect to x is given by

$$-\frac{\partial H}{\partial x} = \sum_{i=0}^{2}\left(1 + \log\frac{x^{y_i}}{A}\right)\frac{y_i x^{y_0-1}A - x^{y_i}A'}{A^2} \tag{11}$$

$$= \sum_{i=0}^{2}\log\frac{x^{y_i}}{A} \times \frac{y_i x^{y_0-1}A - x^{y_i}A'}{A^2} \tag{12}$$

$$= \sum_{i=0}^{2}\left(\log x^{y_i} - \log A\right)\frac{y_i x^{y_0-1}A - x^{y_i}A'}{A^2} \tag{13}$$

$$= \sum_{i=0}^{2}\log x^{y_i} \times \frac{y_i x^{y_0-1}A - x^{y_i}A'}{A^2} \tag{14}$$

$$= \sum_{i=0}^{2}\log x \times \frac{y_i\left(y_i x^{y_0-1}A - x^{y_i}A'\right)}{A^2} \tag{15}$$

where

$$A' = \frac{\partial A}{\partial x} = y_0 x^{y_0-1} + y_1 x^{y_1-1} + y_2 x^{y_2-1}. \tag{16}$$

Note that

$$(y_0 x^{y_0-1} + y_1 x^{y_1-1} + y_2 x^{y_2-1})A = (x^{y_0} + x^{y_1} + x^{y_2})A' \tag{17}$$

and the equalities in (12) and (14) come from (17).

Then, we have

$$-\frac{\partial H}{\partial x}\frac{A^2}{\log x} = \sum_{i=0}^{2} {y_i}^2 x^{y_i-1}A - y_i x^{y_i}A' \tag{18}$$

$$= x^{y_0+y_1-1}(y_0 - y_1)^2 \tag{19}$$

$$+ x^{y_1+y_2-1}(y_1 - y_2)^2 \tag{20}$$

$$+ x^{y_2+y_0-1}(y_2 - y_0)^2 \tag{21}$$

$$\geq 0. \tag{22}$$

Thus, if $x > 1$, then $\frac{\partial H}{\partial x} \leq 0$ and H is a strictly monotonic decreasing function of x unless $y_0 = y_1 = y_2$. Moreover, $x = e^{\beta}$ is a strictly monotonic increasing function of β, and $x > 1$ if $\beta > 0$. Therefore, H is a strictly monotonic decreasing function of β, provided that $\beta > 0$. □

B. Implementation and Training Details

In this section, we describe the implementation and training details of CTC. First, we describe the software for traditional codecs and the libraries for learning-based algorithms. Second, we present the implementation details of the proposed context models CRR and CDR. Then, we explain how to train the proposed CTC algorithm. Note that the implementation and acceleration details of DPICT [27] are available in [22].

B.1. Software and Libraries

We adopt the traditional codecs JPEG2000 [2], BPG444 [10], VTM 12.0 [11] for comparison.

JPEG2000: We use the open software in [2]. We execute the following commands for encoding and decoding. We transform RGB-formatted images, such as png files, into raw files.

```
{buildpath}/opj_compress -i {inputfile}
-o {bin} -r {15:150}
-F {width},{height},3,8,u@1x1:1x1:1x1

{buildpath}/opj_decompress -i {bin} -o
{outputfile}
```

BPG444: We use the software in [10] and enter the following commands.

```
{buildpath}/bpgenc {inputfile} -o {bin}
-q {26:52}
-f 444 -e x265

{buildpath}/bpgdec -o {outputfile} {bin}
```

VTM 12.0: We execute the reference software package in https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/VTM-12.0 with the following commands.

```
{buildpath}/EncoderApp -i {inputfile} -c
{cfgpath}/encoder_intra_vtm.cfg
-o /dev/null -b {bin} -wdt {width} -hgt
{height} -fr 1 -f 1
-q 24, 26, 30, 31:43
--InputChromaFormat=444
--InputBitDepth=8
--ConformanceWindowMode=1
--InputColourSpaceConvert=RGBtoGBR
--SNRInternalColourSpace=1
--OutputInternalColourSpace=0

{buildpath}/DecoderApp -b {bin} -o
{outputfile} -d 8
--OutputColourSpaceConvert=GBRtoRGB
```

We use `Pytorch` [36] and `CompressAI` [8] libraries

to implement the proposed CTC algorithm. Also, we employ the source codes and pretrained parameters in `CompressAI` for the Minnen *et al.*'s algorithm [33]. For the other learning-based codecs, we use the results provided in the original papers.

B.2. Implementation of CRR and CDR

The main network of the proposed CTC algorithm is in Figure 2, and the detailed structures of the CRR and CDR modules are in Figure 4. The context modules are incorporated into the main network as follows. There are three CRR models for different intervals of trit-plane levels l. We denote them as CRR_L, CRR_{L-1}, and $\text{CRR}_{\leq L-2}$, where the subscripts indicate the ranges of trit-plane levels in which the corresponding models are used. In other words,

$$\widetilde{\mathbf{P}}_l = \begin{cases} \text{CRR}_L\left(\widehat{\mathbf{Y}}_{l-1}, \mathbf{M}, \mathbf{\Sigma}, \mathbf{E}_l, \mathbf{P}_l\right) & \text{if } l = L, \\ \text{CRR}_{L-1}\left(\widehat{\mathbf{Y}}_{l-1}, \mathbf{M}, \mathbf{\Sigma}, \mathbf{E}_l, \mathbf{P}_l\right) & \text{if } l = L-1, \\ \text{CRR}_{\leq L-2}\left(\widehat{\mathbf{Y}}_{l-1}, \mathbf{M}, \mathbf{\Sigma}, \mathbf{E}_l, \mathbf{P}_l\right) & \text{if } l \leq L-2. \end{cases} \tag{23}$$

Similarly, we implement three CDR models CDR_{L-1}, CDR_{L-2}, and $\text{CDR}_{\leq L-3}$ to obtain

$$\widetilde{\mathbf{Y}}_l = \begin{cases} \widehat{\mathbf{Y}}_l & \text{if } L-1 < l \leq L, \\ \text{CDR}_{L-1}\left(\widehat{\mathbf{Y}}_l, \mathbf{M}, \mathbf{\Sigma}\right) & \text{if } L-2 < l \leq L-1, \\ \text{CDR}_{L-2}\left(\widehat{\mathbf{Y}}_l, \mathbf{M}, \mathbf{\Sigma}\right) & \text{if } L-3 < l \leq L-2, \\ \text{CDR}_{\leq L-3}\left(\widehat{\mathbf{Y}}_l, \mathbf{M}, \mathbf{\Sigma}\right) & \text{if } l \leq L-3. \end{cases} \tag{24}$$

Whereas CRR estimates the probability tensor $\widetilde{\mathbf{P}}_l$ for each trit-plane $\mathbf{T}_l$ ($l = 1, \ldots, L$), CDR performs the prediction of $\widetilde{\mathbf{Y}}_l$ for any $l \leq L-1$. Therefore, l is an integer in (23) but a real number in (24). The trit-plane levels $l \leq L-2$ are supported by a single CRR model, and the levels $l \leq L-3$ are by a single CDR model. These choices are made to strike a balance between the number of parameters and the RD performance. Also, CDR is not used at the top level $L-1 < l \leq L$ because the refinement of a latent tensor $\widehat{\mathbf{Y}}_l$ is not necessary at such a fine level.

Note that the proposed CDR is conceptually similar to LRP in [34]. However, there are clear differences between them. Whereas LRP predicts residual errors by taking only the mean and latent tensors as input, CDR exploits the standard deviation $\mathbf{\Sigma}$ as the additional context. In this way, CDR can refine partially reconstructed latent elements by exploiting their uncertainty levels, which are inversely proportional to the standard deviations. To demonstrate the importance of $\mathbf{\Sigma}$, we have implemented an ablated version of CDR without $\mathbf{\Sigma}$. It increases the BD-rate by $+2.45\%$ on the Kodak lossless dataset. Moreover, while the quantization step size is 1 in LRP, it is larger in the proposed algorithm (*i.e.* 3^{L-l} when the first l trit-planes are decoded). Thus, there are more opportunities for reducing quantization errors in the proposed algorithm. To achieve this goal, the proposed CDR exploits both $\mathbf{M}$ and $\mathbf{\Sigma}$.

B.3. Training of CTC

We train the main network for 300 epochs using the rate-distortion loss $\mathcal{L} = \mathcal{D} + \lambda\mathcal{R}$ with $\lambda = 5$.

In the trit-plane slicing module in Figure 2, a latent tensor is sliced into L trit-planes. Note that the maximum trit-plane level L depends on the latent tensor, as described in [22]. However, $L = 7$ for most images. The selection of CRR and CDR models in (23) and (24) is dependent on L. Therefore, for stable training of these models, as well as the decoder retraining, we fix $L = 7$ and use the training images with $L = 7$ only.

We use the cross-entropy loss in (4) to train the three CRR models. The CRR process is performed for every trit, except when the original probabilities are $(p_0, p_1, p_2) = (0, 1, 0)$. In such a case, the trit requires no bit, and there is no reason to update its probabilities.

The CDR loss in (6) can be rewritten as

$$\ell_{\text{CDR}}(l) = \|\mathbf{Y} - \widetilde{\mathbf{Y}}_l\|_F \tag{25}$$

where l denotes a trit-plane level. The first CDR model CDR_{L-1} in (24) supports a partially reconstructed level $l \in (L-2, L-1]$. For its training, we use the sum of losses, given by

$$\ell_{\text{CDR}}(L-1) + \ell_{\text{CDR}}(\alpha) + \ell_{\text{CDR}}(L-2) \tag{26}$$

where $\alpha \sim \mathcal{U}(0,1)$ is a uniform random variable. The losses for the other two models CDR_{L-2} and $\text{CDR}_{\leq L-3}$ are similarly defined.

Then, the decoder is retrained to minimize the loss ℓ_{DEC} in (7). Note that the original decoder is optimized for the case when all trit-planes are received (*i.e.* the highest level $l = L$). Thus, the decoder is retrained to consider lower levels as well. However, due to the retraining, the performances at high levels can be degraded. To alleviate the degradation, we set large weighting parameters at high levels, compared to low levels. Specifically, we define the loss as

$$\ell_{\text{DEC}} = 100\times \textstyle\sum_{l=L-1}^{L} \|g_s(\widetilde{\mathbf{Y}}_l) - \mathbf{X}\|_F \tag{27}$$

$$+ \textstyle\sum_{l=L-4}^{L-2} \|g_s(\widetilde{\mathbf{Y}}_l) - \mathbf{X}\|_F. \tag{28}$$

Note that we consider five levels from $L-4$ to L, and set bigger weights at the two highest levels L and $L-1$.

The training epochs for the context models and the decoder retraining are summarized in Table 4. These training schedules are determined by observing the convergence of the validation performance.

Table 4. The numbers of epochs for the context model training and the decoder retraining (g_s).

CRR_L	CRR_{L-1}	$\text{CRR}_{\leq L-2}$	CDR_{L-1}	CDR_{L-2}	$\text{CDR}_{\leq L-3}$	g_s
300	300	10	30	30	30	100

C. More Experiments

C.1. RD curves

Figures 12 and 13 compare the RD curves on the CLIC validation dataset and the JPEG-AI testset, respectively. All learning-based algorithms, including the proposed CTC, are optimized to minimize the MS-SSIM loss in Figure 12(b) and Figure 13(b).

Figure 14 compares the proposed CTC with the trit-plane coding without RD priorities. More specifically, in 'Without RD priorities,' the trits in each trit-plane are transmitted in the 3D raster scan order, instead of the decreaing order of their RD priorities [27]. This alternative method performs badly compared to CTC. However, we see that its performance is also improved by employing the two context modules, CRR and CDR, and the decoder retraining scheme.

C.2. Time complexity for high-resolution images

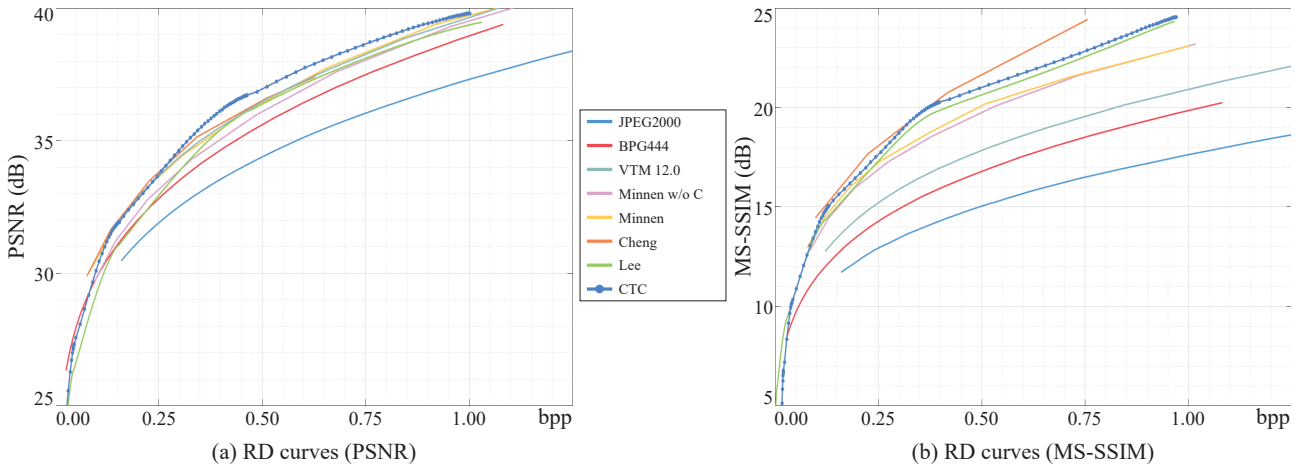
We compare the time complexities for compressing 2K images in the CLIC validation dataset and the JPEG-AI testset in Table 5. The proposed CTC algorithm is based on trit-plane coding, which represents each latent element with about 7 trits. Hence, CTC increases the number of entropy-coded data by a factor of about 7, as compared with non-FGS codecs such as He *et al*. [20]. This is the main reason (and a price for enabling FGS) that CTC is slower than [20]. However, it can be observed from Table 5 that the proposed context modules, CRR and CDR, increase the complexities only moderately.

Table 5. Time complexity comparison of CTC with Minnen *et al*. [33] and He *et al*. [20].

	Encoding (s)	Decoding (s)
He *et al*. [20]	1.00	0.91
Minnen *et al*. [33]	25.85	78.49
CTC w/o context modules	8.10	7.19
CTC	8.70	8.26

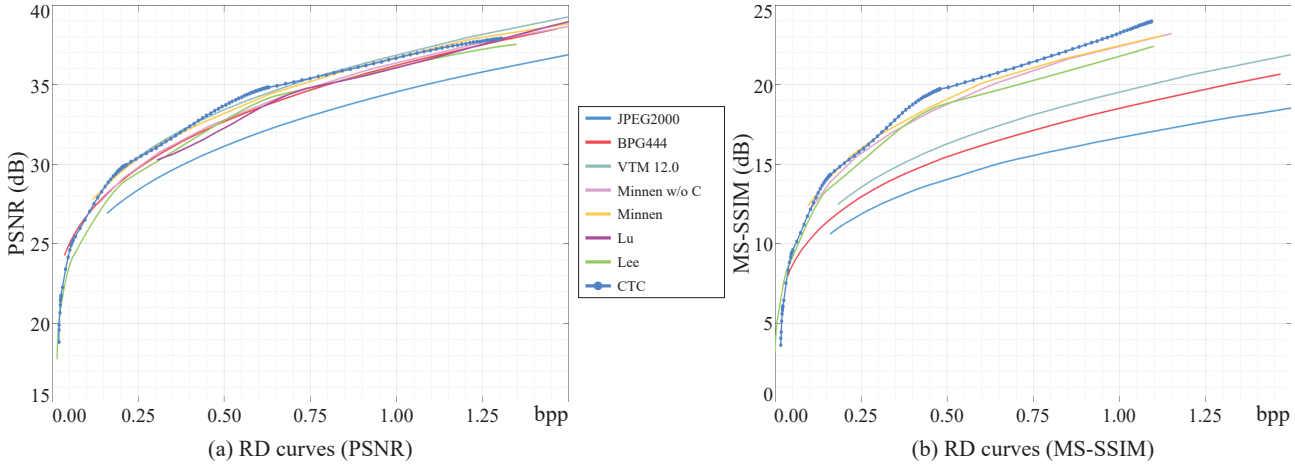
C.3. Reconstructed images of CTC

Figures 15∼20 show various images reconstructed by the proposed CTC algorithm at levels $l = L$, $L-2$, $L-3$, and $L-4$. The images with resolutions larger than 512×768 are center-cropped.



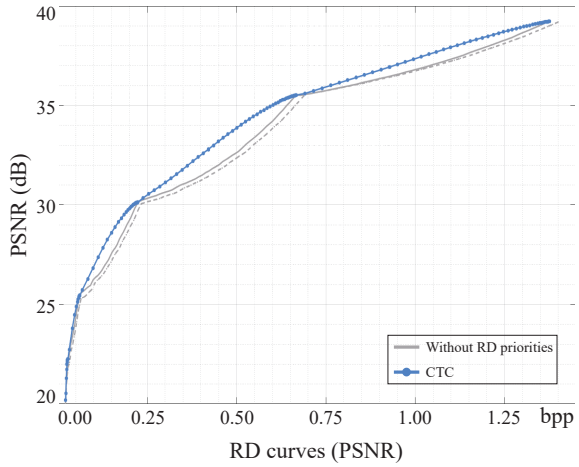
(a) RD curves (PSNR) (b) RD curves (MS-SSIM)

Figure 12. RD curve comparison on the CLIC validation dataset.



(a) RD curves (PSNR) (b) RD curves (MS-SSIM)

Figure 13. RD curve comparison on the JPEG-AI testset.



RD curves (PSNR)

Figure 14. RD curve comparison of CTC and the alternative trit-plane coding method without RD priorities on the Kodak lossless dataset. The dashed curve means that the context modules, CRR and CDR, and the decoder retraining are not employed.

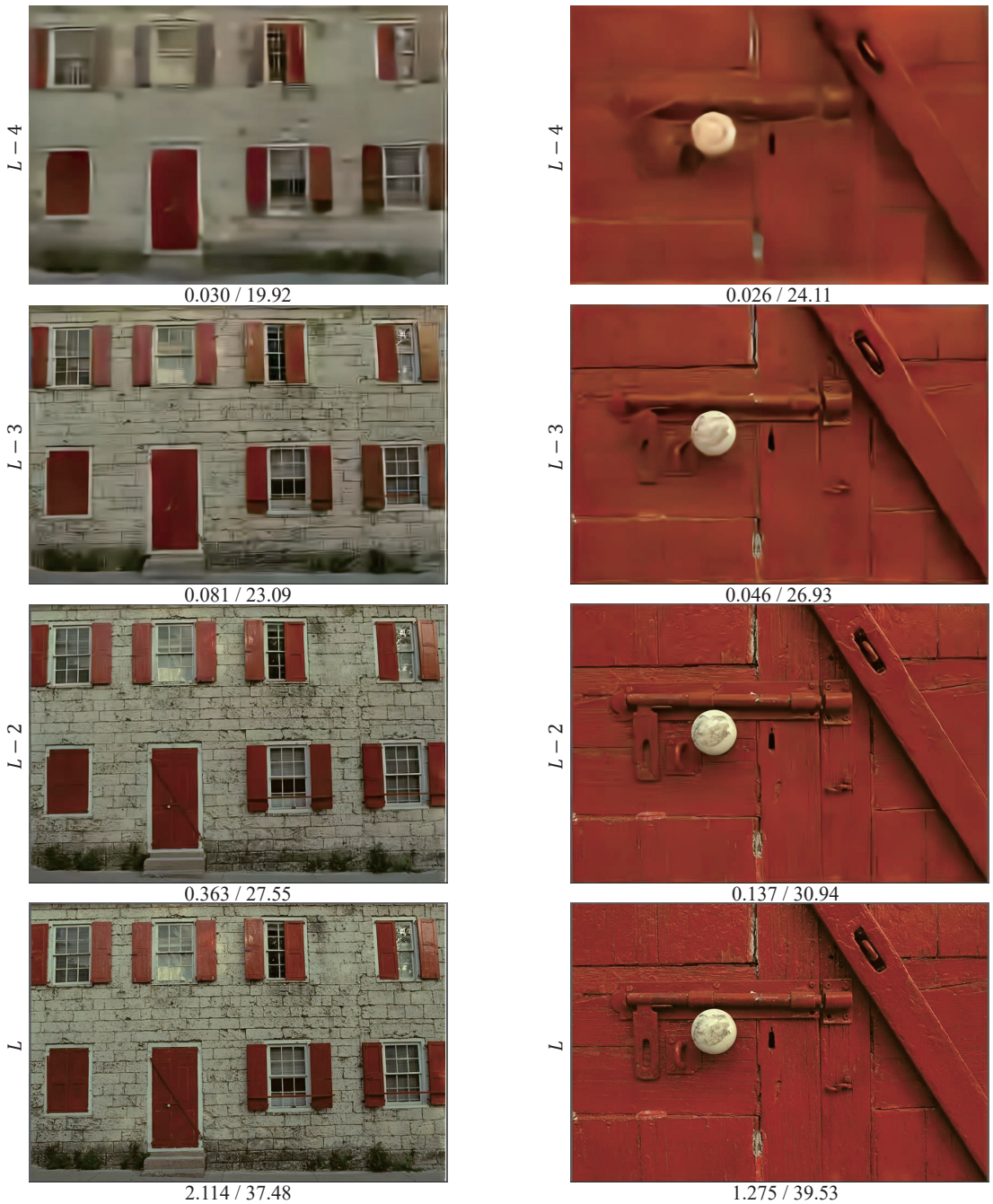


Figure 15. Reconstructed images of "kodim01.png" and "kodim02.png." The bitrates (bpp) and PSNRs (dB) are reported below each image.

Figure 16. Reconstructed images of "kodim09.png" and "kodim19.png." The bitrates (bpp) and PSNRs (dB) are reported below each image.

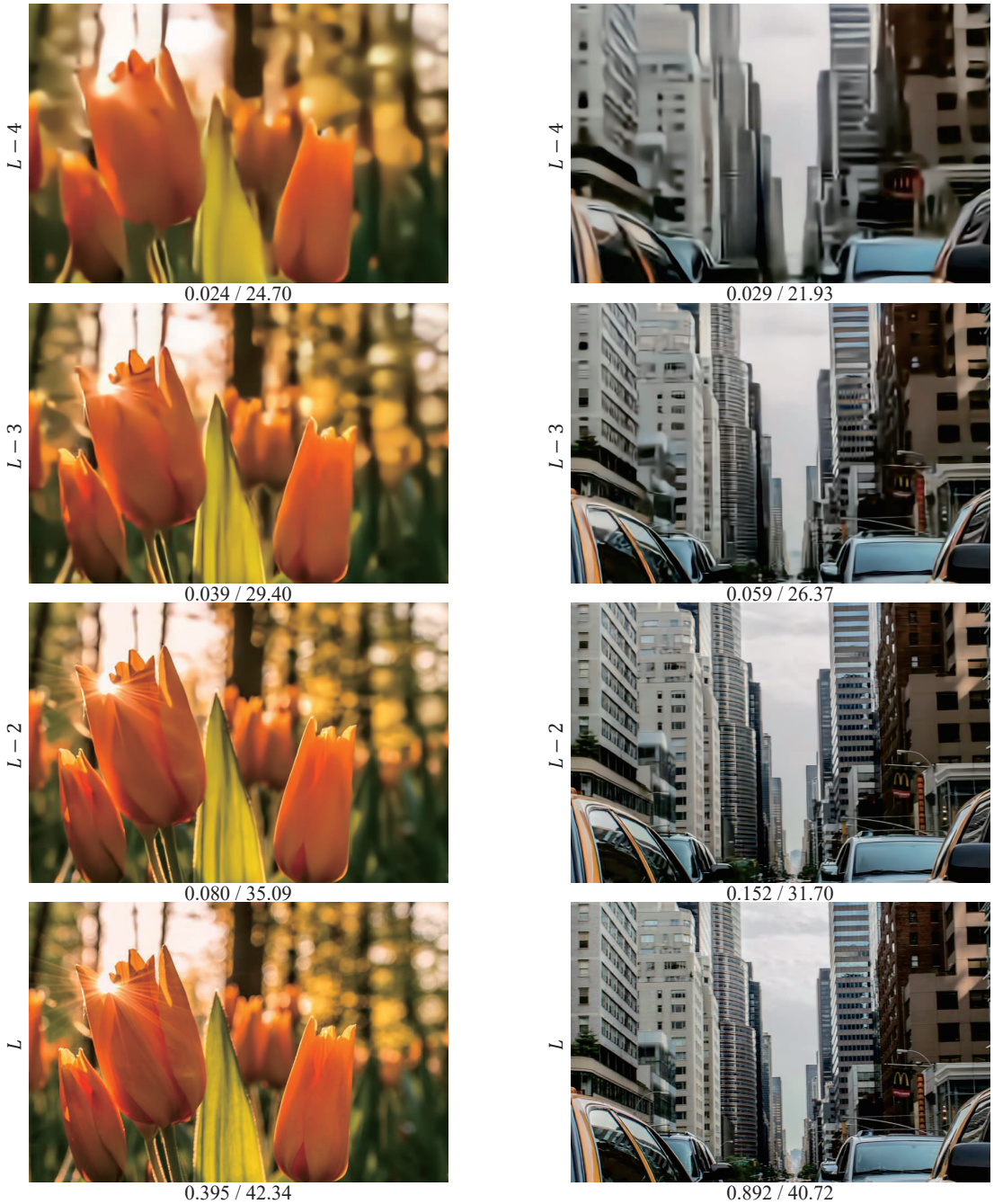


Figure 17. Reconstructed images of “ales-krivec15949.png” and “andrew-ruiz-376.png.” The bitrates (bpp) and PSNRs (dB) are reported below each image.

Figure 18. Reconstructed images of “nomao-saeki-33553.png” and “philipp-reiner-207.png.” The bitrates (bpp) and PSNRs (dB) are reported below each image.

Figure 19. Reconstructed images of "000505_TE_1336x872.png" and "000505_TE_1336x872.png." The bitrates (bpp) and PSNRs (dB) are reported below each image.

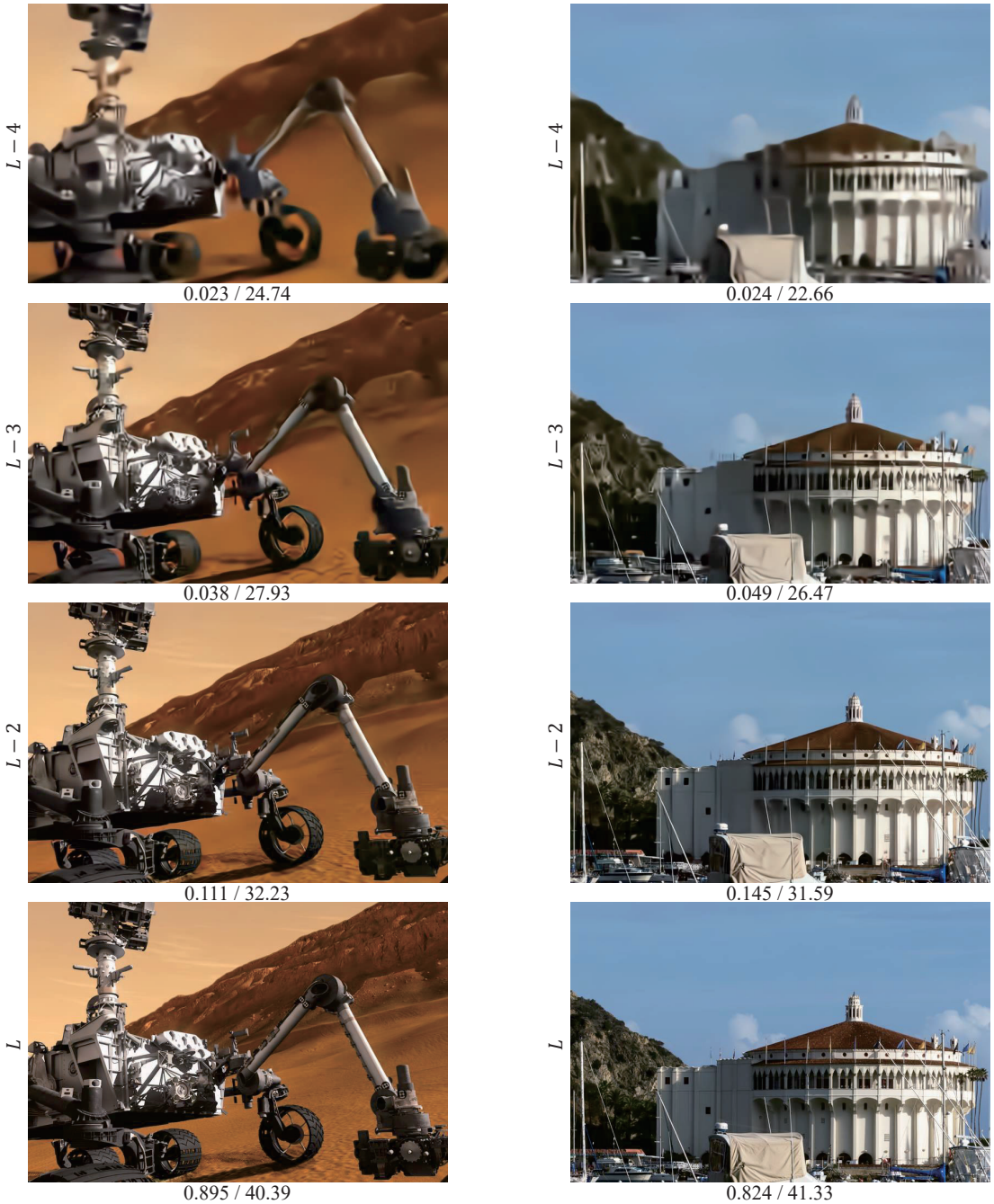


Figure 20. Reconstructed images of "00010_TE_2000x1128.png" and "00015_TE_3680x2456.png." The bitrates (bpp) and PSNRs (dB) are reported below each image.