Supplemental Document on

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

S-1. Implementation

Compression network: We develop the compression network similar to the Cheng *et al*.'s network [11], but we make some modifications to encode scalable bitstreams. Figure S-1 shows the architecture of the compression network, and Figure S-2 shows the detailed structure of the attention module that is simplified as a green block in Figure S-1.

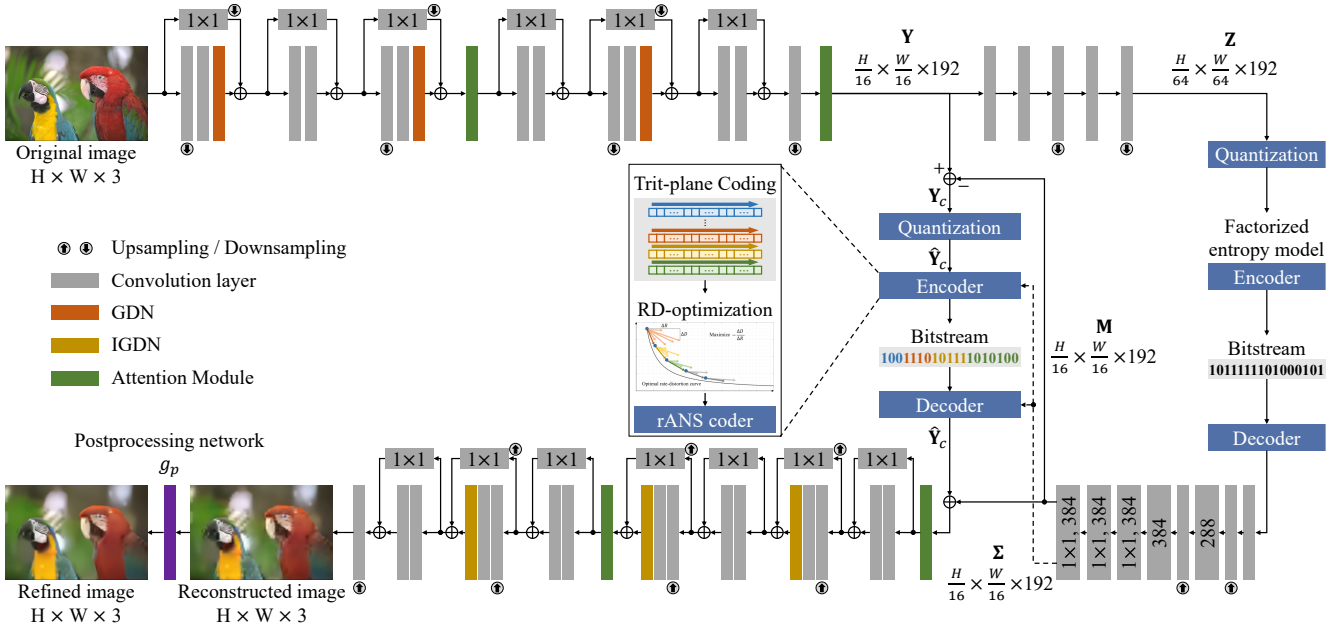


Figure S-1. Architecture of the compression and postprocessing networks, in which layers are shown in different colors depending on their types. A gray block is a convolution layer, in which the kernel size and the number of channels are specified unless they are the default parameters of 3×3 and 192, respectively. A downward arrow indicates that the convolution layer performs downsampling with a stride of 2, while an upward arrow indicates a subpixel convolution layer with an upsample factor of 2. GDN and IGDN are a generalized divisive normalization layer and its inverse version [S1]. The detailed structure of the attention module, depicted by a green block, is shown in Figure S-2, and the postprocessing network g_p, depicted by a purple block, is shown in Figure S-4.

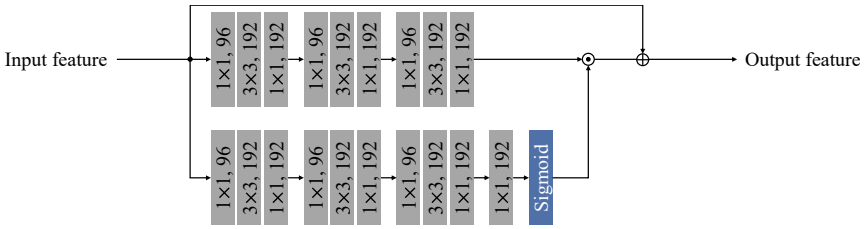


Figure S-2. Network structure of the attention module, where $\odot$ indicates the element-wise product operation.

RD-prioritized transmission: In Figure 10 in the main paper, the same bitstream of DPICT for an image is reconstructed at 164 different rates. We apply the scalable coding from the $(L-5)$th trit-plane to the Lth trit-plane. Since more significant trit-planes (*i.e.* the nth trit-planes, $n \leq L-6$) produce a very short bitstream in general, we encode them together as a coding unit. Then, starting from the $(L-5)$th trit-plane, all trits in each trit-plane are divided into 48 bins according to the RD priorities. We find experimentally that the encoding of low-priority bins reduces the distortion only marginally. Therefore, we encode those bins together as a coding unit. Figure S-3 shows which bins are grouped into a coding unit. For example, in the $(L-2)$th trit-plane, the 17 lowest-priority bins are encoded together, and thus 32 different coding units are available. By summing up the number of coding units in each trit-plane, we have $164 = 1+3+16+16+32+48+48$ different rates.

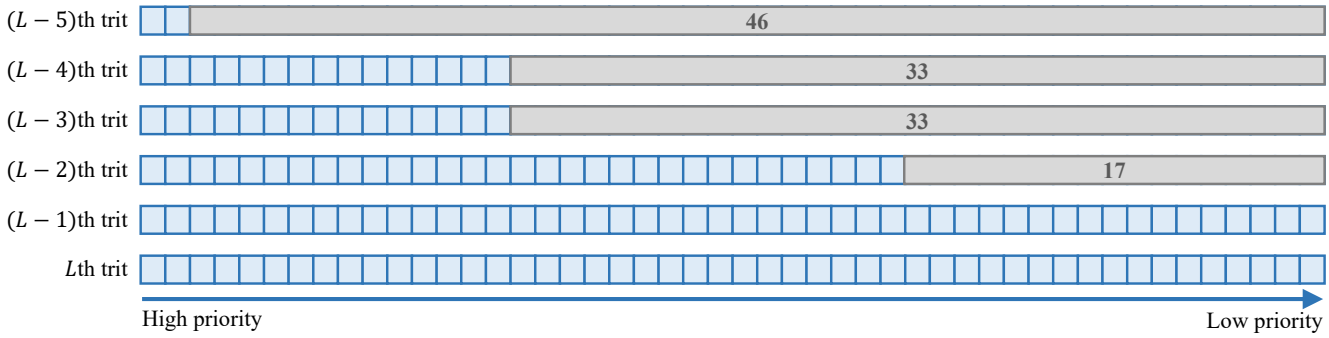


Figure S-3. Dividing a trit-plane into coding units: each trit-plane is divided into 48 bins, and the low-priority bins in a gray block is grouped into a coding unit.

Postprocessing networks: Figure S-4 shows the detailed architecture of the postprocessing networks g_p. We train two postprocessing networks, targeting at different bit-rates: The first g_p targets at $\hat{\mathbf{X}}^n$ for $n \in [0, L-2.9]$, and the second g_p for $n \in (L-2.9, L-1.8]$. To train the two post-processing networks, n in (20) is set to $L-3$ and $L-2$, respectively.

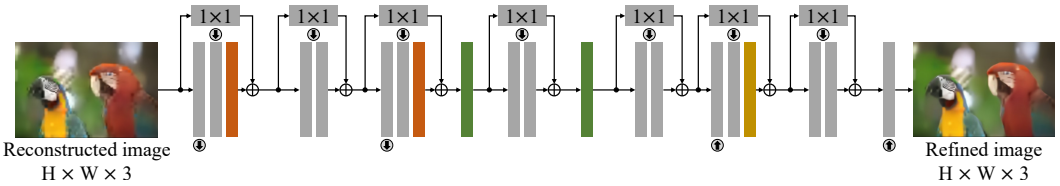


Figure S-4. Network structure of the postprocessing networks g_p.

S-2. Computational Complexity

We compare DPICT with Minnen *et al.* [30] and Cheng *et al.* [11] in terms of runtime and FLOPs in Table S-1. The experiments are done with an AMD Ryzen 9 3900X CPU and an NVIDIA GeForce RTX 3090 GPU. Note that we develop DPICT based on the Cheng *et al.*'s network. For DPICT, the encoding is done only once for all 164 rates in Figure 10, and the decoding complexity is averaged over the 164 rate runs. For both encoding and decoding, DPICT is faster than [11], although it requires similar FLOPs. The autoregressive convolution in [30] and [11] is time-consuming, while the trit-plane coding is done efficiently and supports GPU parallel computing. Figure S-5 plots the decoding times for the 164 rates. Even at the highest rate, DPICT can reconstruct an image within 6 seconds.

Table S-1. Runtime and GFLOPs comparison of the proposed DPICT with conventional codecs on the Kodak dataset.

	Minnen *et al.* [30]		Cheng *et al.* [11]		DPICT		
	Encoding	Decoding	Encoding	Decoding	Encoding	Decoding	Postprocessing
Runtime (s)	4.452	11.47	3.035	8.695	1.108	2.728	0.051
GFLOPs	35.12	133.6	159.1	228.9	154.2	224.0	333.6

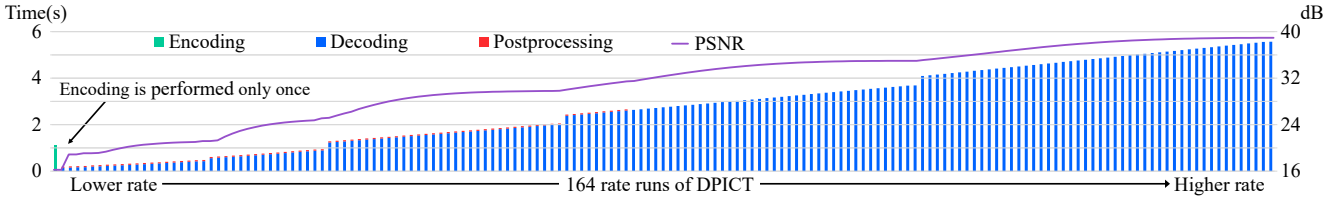


Figure S-5. Running time of the proposed DPICT algorithm for the 164 rates on the Kodak dataset.

S-3. Postprocessing Network Complexity

As the tiny red bars in Figure S-5 indicate, the postprocessing demands only a small fraction of the runtime; most of the runtime is spent on the trit-plane decoding. In Figure S-6, we compare the postprocessing runtimes, FLOPs, and PSNR gains, when the number of channels of the postprocessing network is reduced by ratios of 0.75, 0.50, and 0.25, respectively.

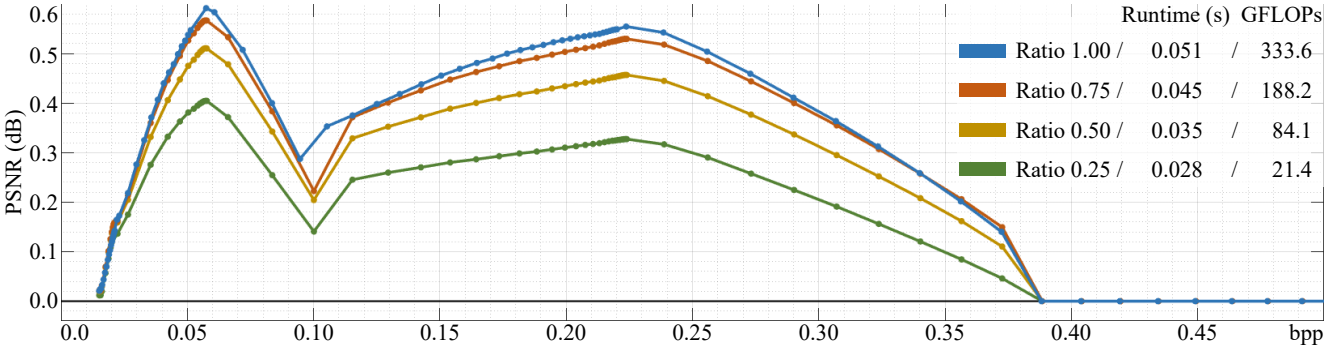
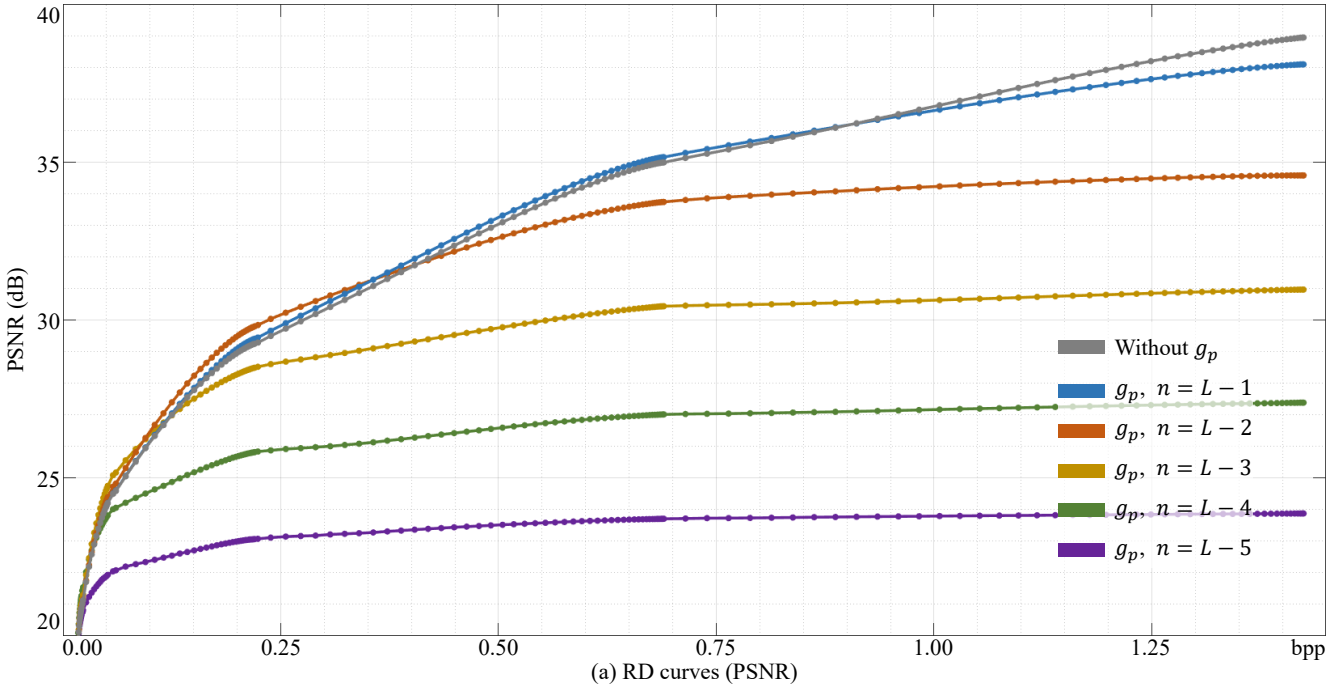


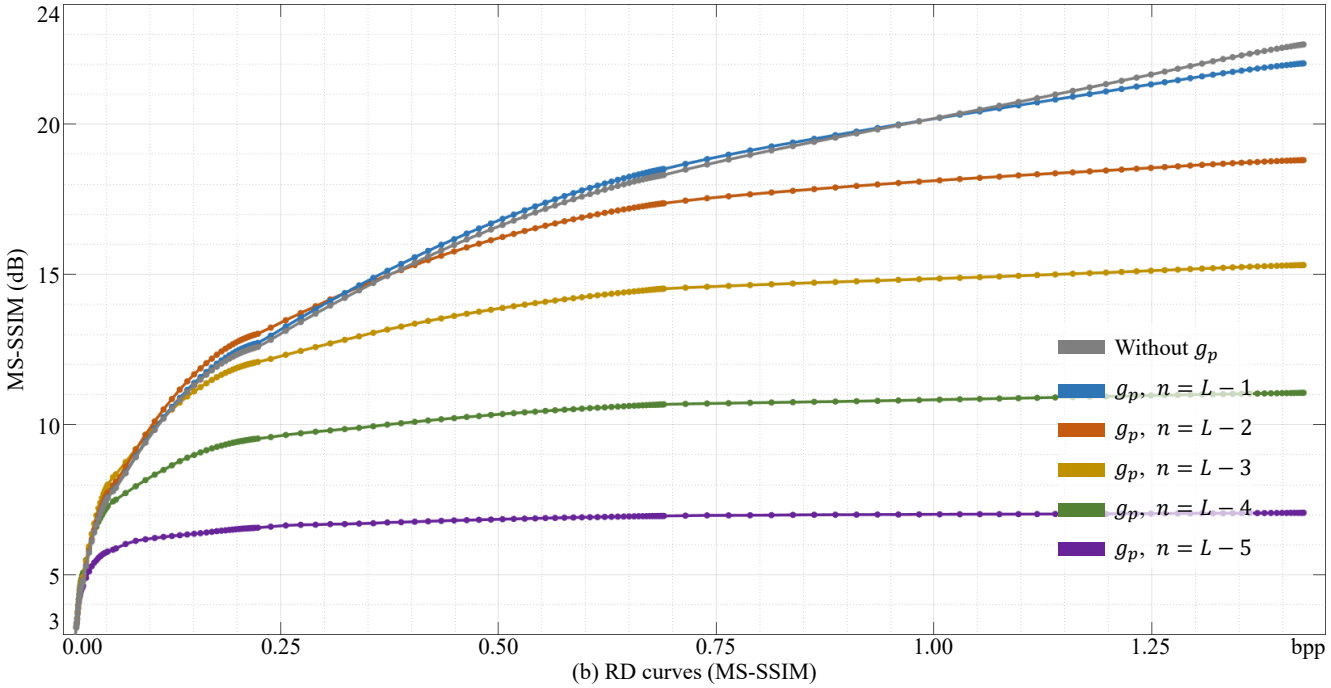
Figure S-6. Performance gains of postprocessing networks according to their complexities on the Kodak dataset. Runtimes and GFLOPs of the postprocessing networks are specified at the top right corner.

S-4. Additional Postprocessing Networks

In the default mode, we train two postprocessing networks g_p, targeting at $\hat{\mathbf{X}}^n$ for $n \in [0, L-2.9]$ and for $n \in (L-2.9, L-1.8]$, respectively. To train the two g_p, we use $\hat{\mathbf{X}}^{L-3}$ and $\hat{\mathbf{X}}^{L-2}$ by setting n to $L-3$ and $L-2$ in (20). Additionally, we train three more g_p using $\hat{\mathbf{X}}^{L-5}$, $\hat{\mathbf{X}}^{L-4}$, and $\hat{\mathbf{X}}^{L-1}$. Figure S-7 compares the RD curves of the five postprocessing networks with those of the baseline without using any postprocessing network. We see that each postprocessing network improves image qualities over a limited range of rates only.



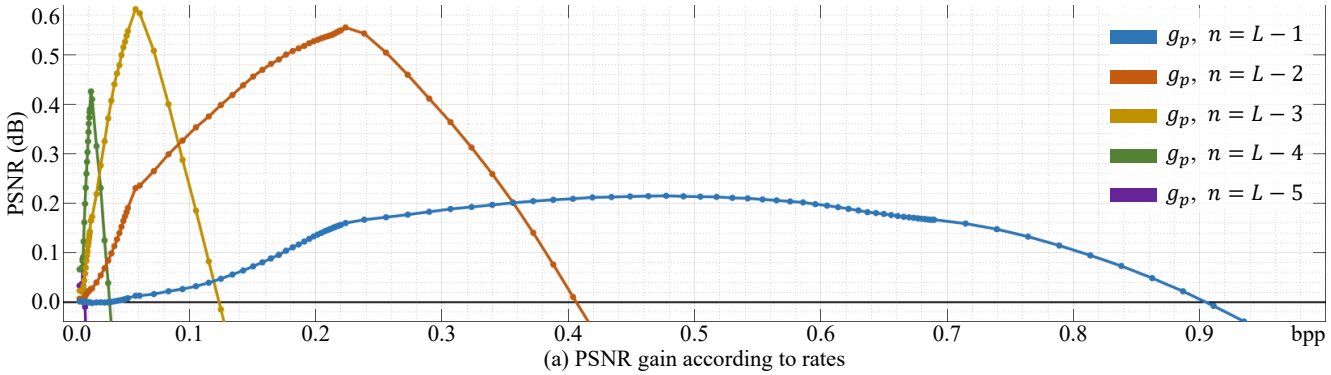
(a) RD curves (PSNR)



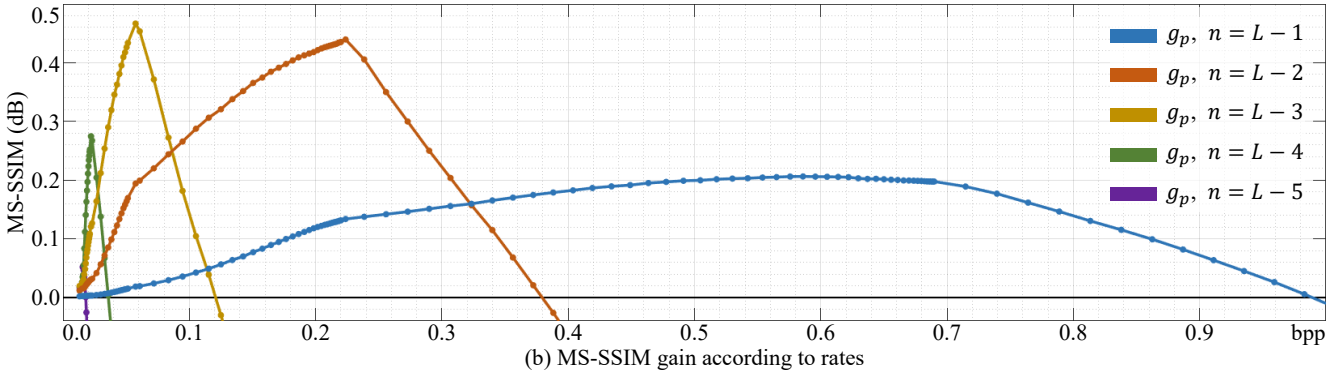
(b) RD curves (MS-SSIM)

Figure S-7. RD performance comparison of the five postprocessing networks on the Kodak dataset.

Figure S-8 plots the PSNR and MS-SSIM gains of each postprocessing network in comparison with the baseline 'Without g_p' in Figure S-7. The postprocessing network trained with $\hat{\mathbf{X}}^{L-1}$ improves the performances over a wide range of rates, but the improvement is limited to maximum 0.2 dB in PSNR. The network trained with $\hat{\mathbf{X}}^{L-4}$ is only applicable to a narrow range of rates, and the network trained with $\hat{\mathbf{X}}^{L-5}$ improves the performance negligibly over an even narrower range. In contrast, the two networks trained with $\hat{\mathbf{X}}^{L-2}$ and $\hat{\mathbf{X}}^{L-3}$ improve image qualities meaningfully over relatively wide ranges. This is why we use these two networks in the default mode.



(a) PSNR gain according to rates

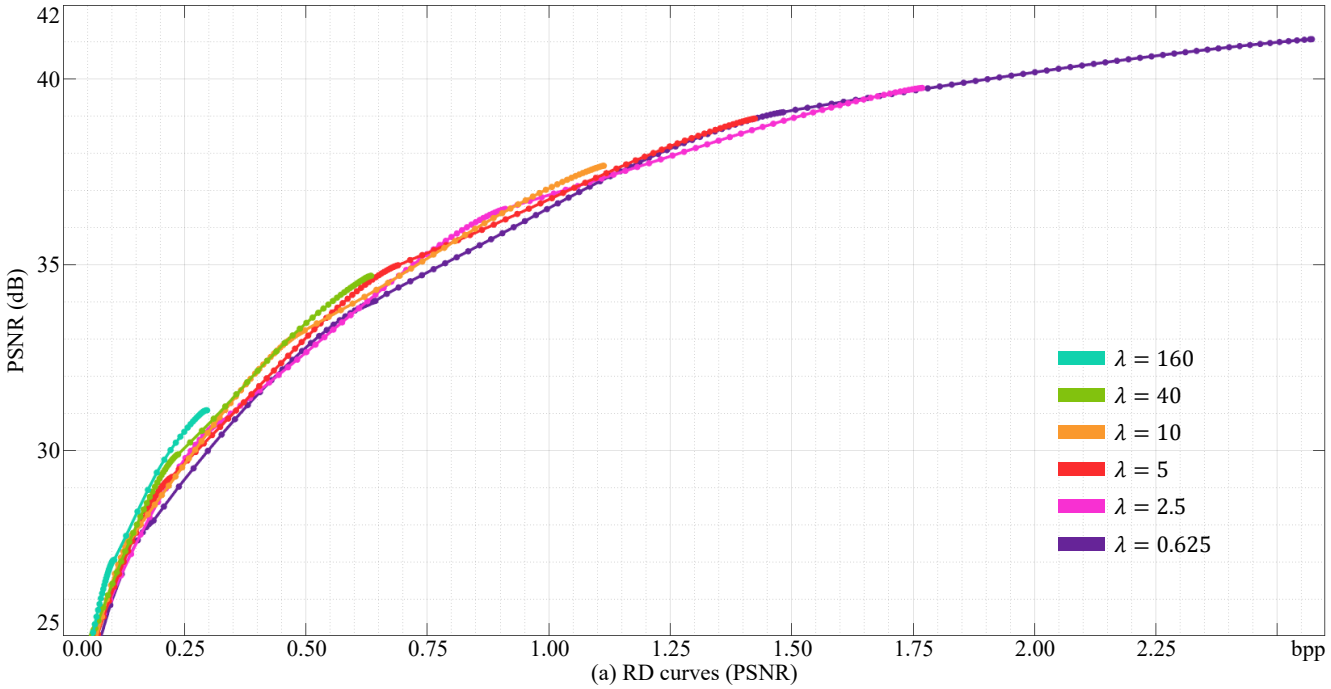


(b) MS-SSIM gain according to rates

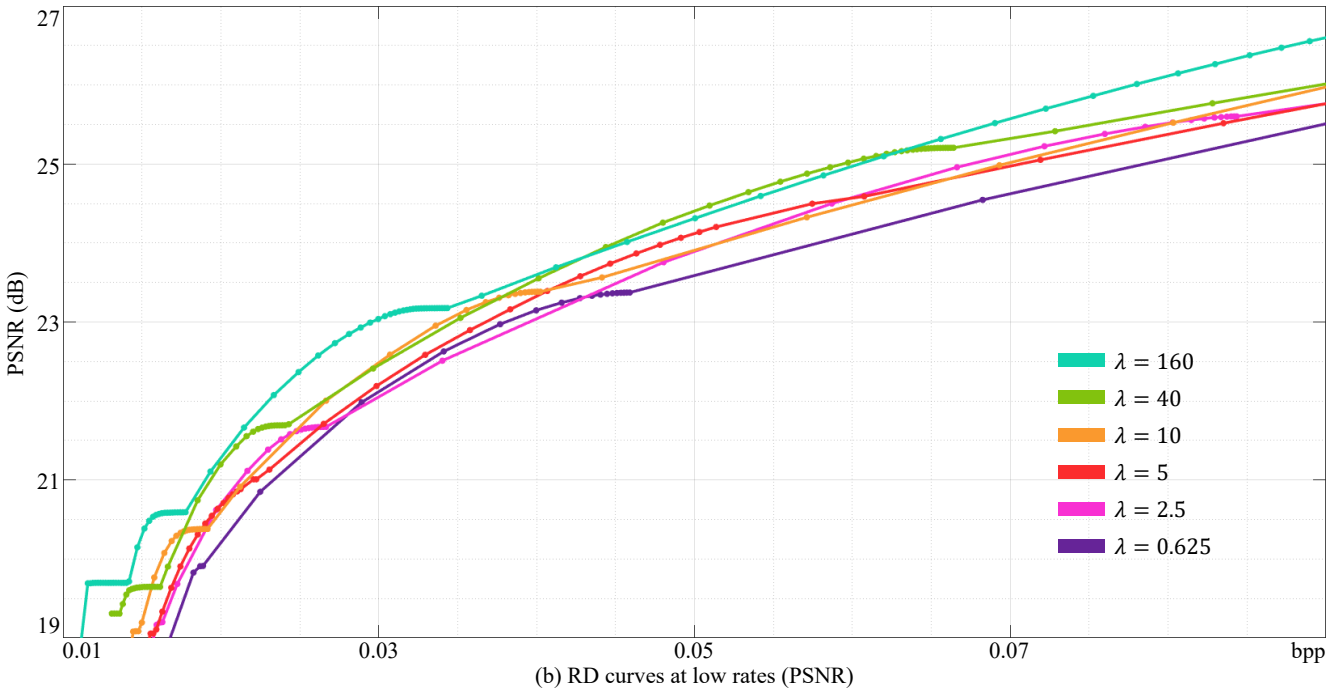
Figure S-8. Performance gains of each postprocessing network in comparison with the baseline on the Kodak dataset.

S-5. Lagrangian Multiplier λ

In the default mode, we set $\lambda = 5$ in the loss function in (5). We also train the compression network using $\lambda \in \{160, 40, 10, 2.5, 0.625\}$ and compare the RD performances in Figure S-9. In this test, the postprocessing networks are not used. Regardless of λ, the proposed DPICT supports FGS. However, a smaller λ can support scalability over a wider range of rates in general. We select $\lambda = 5$ because of its excellent RD performance over a relatively wide range of rates.



(a) RD curves (PSNR)



(b) RD curves at low rates (PSNR)

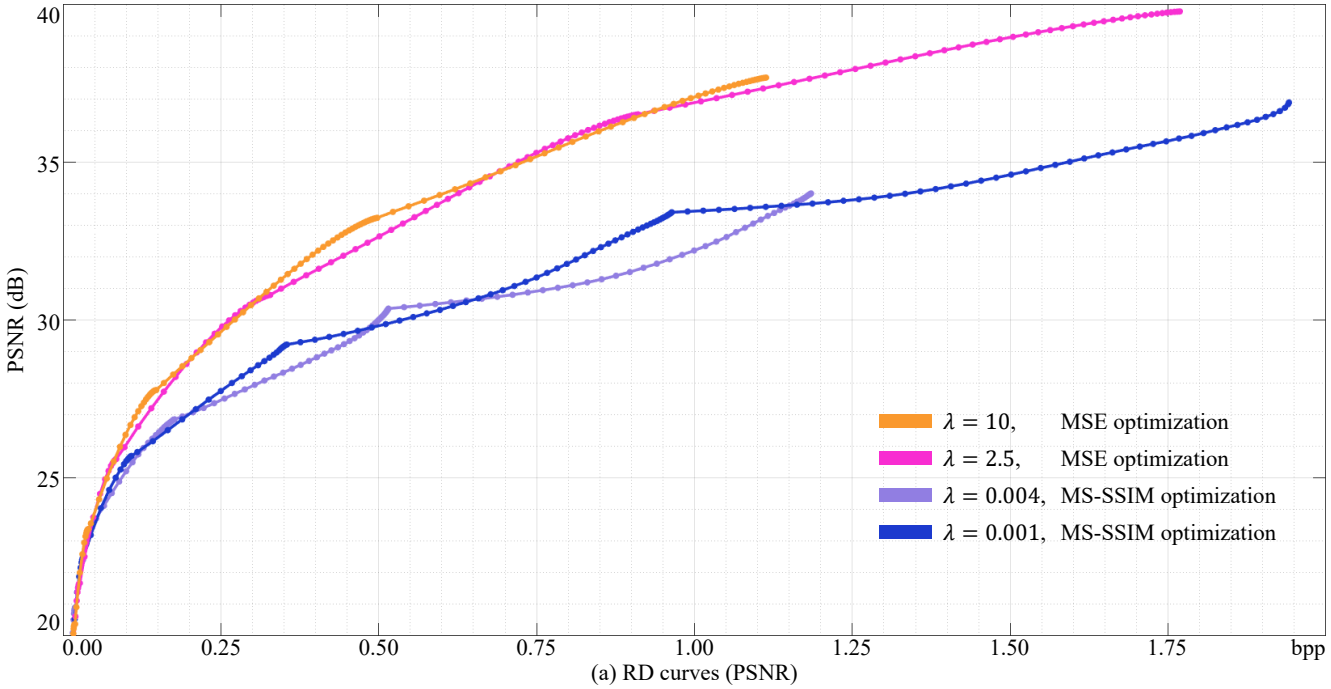
Figure S-9. RD performance comparison according to various settings of λ on the Kodak dataset.

S-6. Training Using MS-SSIM Loss

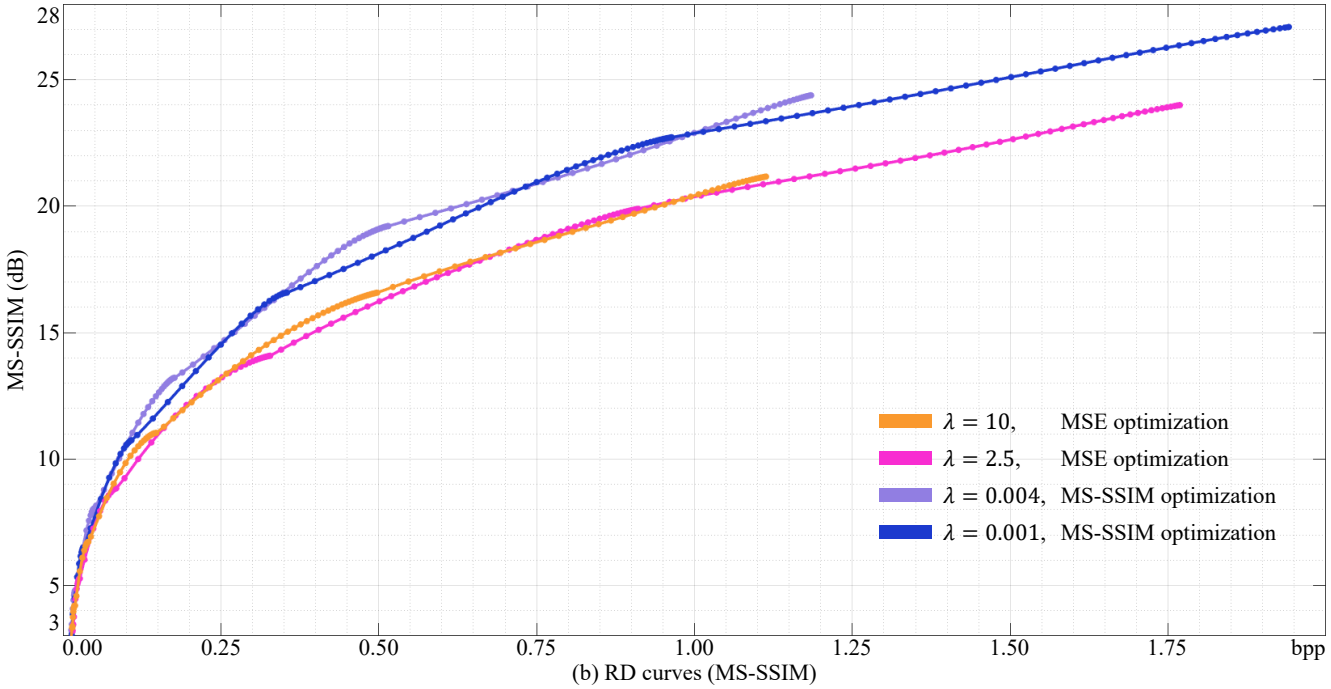
As done in [6], we train the compression network by replacing the distortion term ℓ_D of the loss function in (5) with the MS-SSIM loss

$$\ell_D(\mathbf{X}, \hat{\mathbf{X}}) = (1 - \text{MS-SSIM}(\mathbf{X}, \hat{\mathbf{X}})).$$

Figure S-10 compares the RD performances of networks optimized for the MS-SSIM loss and for the mean squared error (MSE) loss, respectively. In this test, the postprocessing networks are not used. As expected, the models optimized with the MS-SSIM loss provide better MS-SSIM performances than those optimized with the MSE loss.



(a) RD curves (PSNR)



(b) RD curves (MS-SSIM)

Figure S-10. RD performance comparison of models trained with the MS-SSIM loss and with the MSE loss, respectively.

S-7. Different Compression Networks

The proposed DPICT algorithm can also be applied to other compression networks. To verify this, we conduct experiments using two more compression networks: Ballé *et al.*'s network [6] and Minnen *et al.*'s network [30]. For Minnen *et al.*'s network, we remove the autoregressive module because of the reason specified in Section 4.1 in the main paper. For each network, we use the implementation of the compressAI library [7]. Figure S-11 shows the results. Each RD curve of the existing networks, depicted by dashed lines, is obtained by several fixed-rate models. On the contrary, each RD curve of DPICT, in a solid line, is obtained by a single model. Note that DPICT supports FGS at the cost of only slight RD performance degradation.

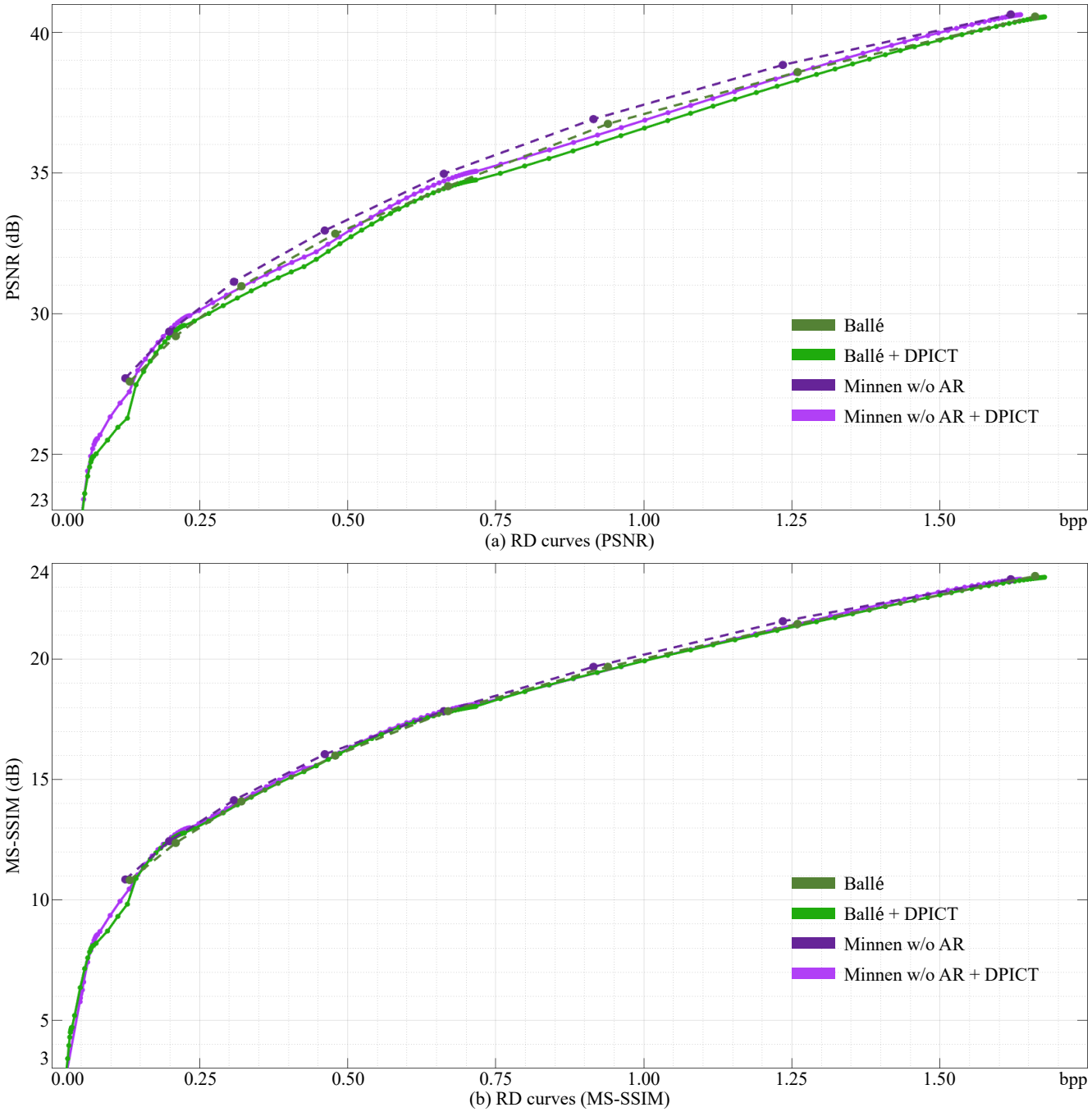


Figure S-11. DPICT is applied to Ballé *et al.*'s network [6] and Minnen *et al.*'s network [29]. In contrast to these existing networks, DPICT supports FGS at the cost of only slight performance degradation.

S-8. More Qualitative Results

- Figures S-12∼S-14 compare reconstructed images at similar rates.
- Figures S-15∼S-21 show images that are reconstructed progressively from a single bitstream.
- Figure S-22 and Figure S-23 show the impacts of the postprocessing networks g_p. For easier comparison, improvement maps are also provided.

Figure S-12. Qualitative comparison of reconstructed images at similar rates in the CLIC dataset.

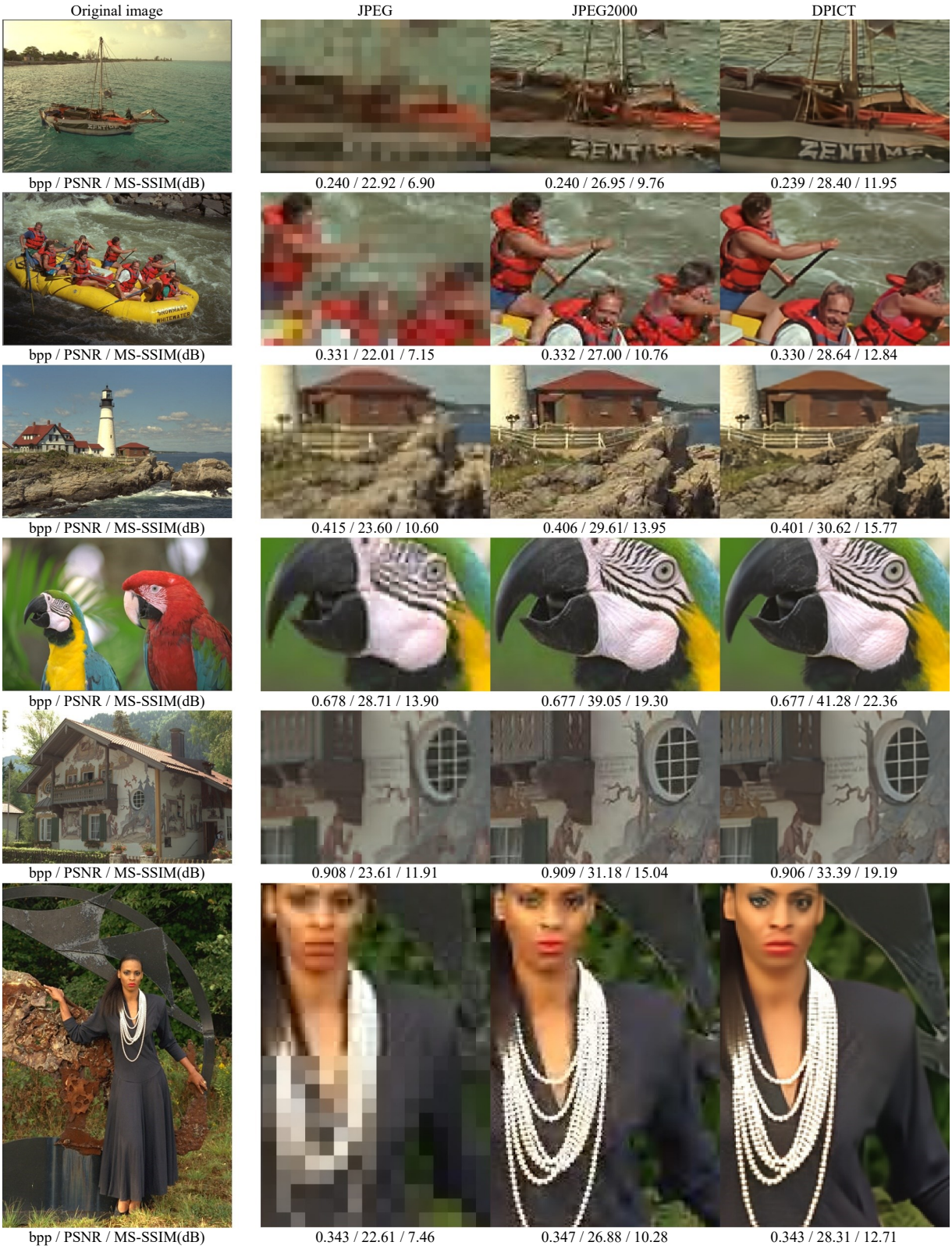


Figure S-13. Qualitative comparison of reconstructed images at similar rates in the Kodak dataset.

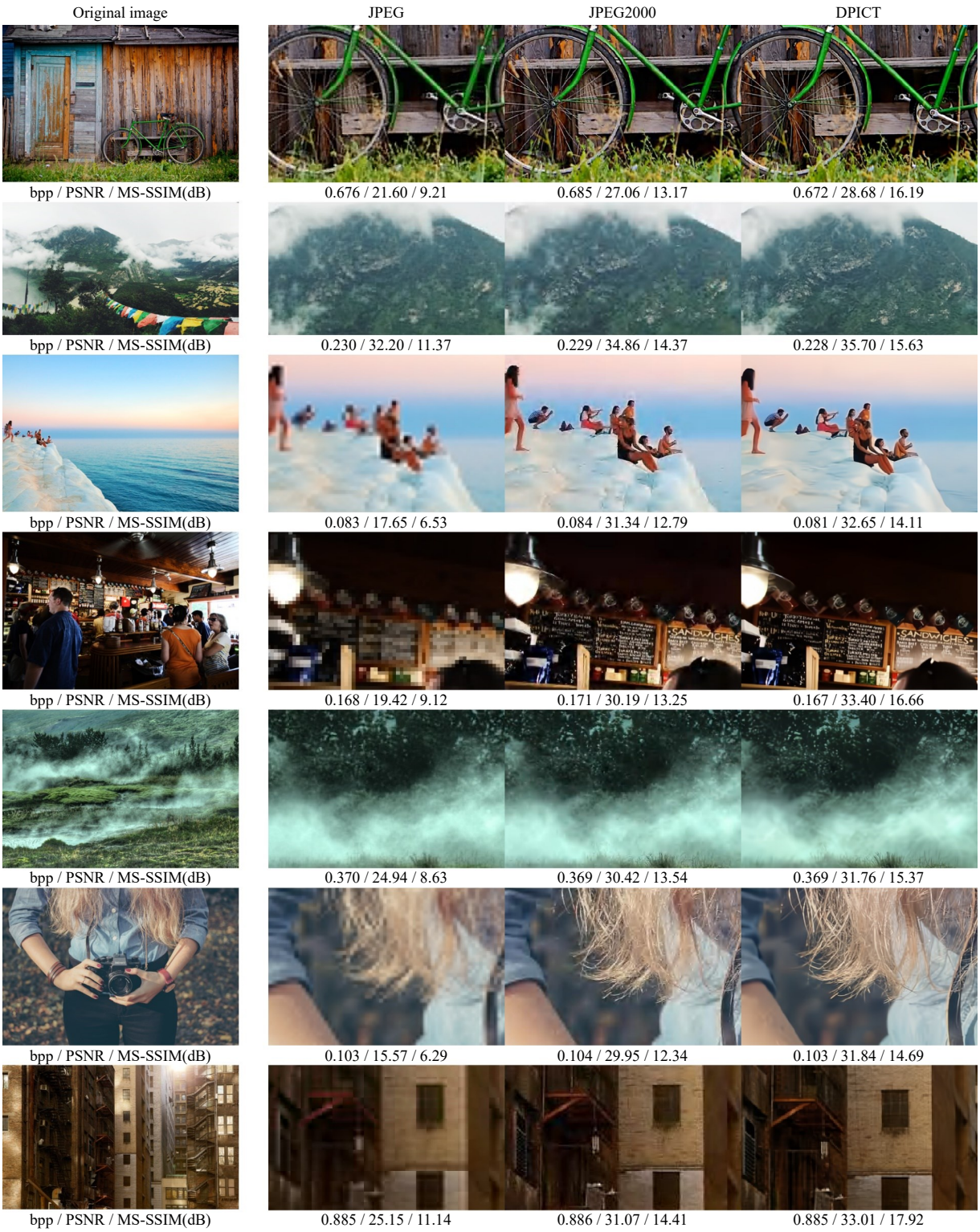
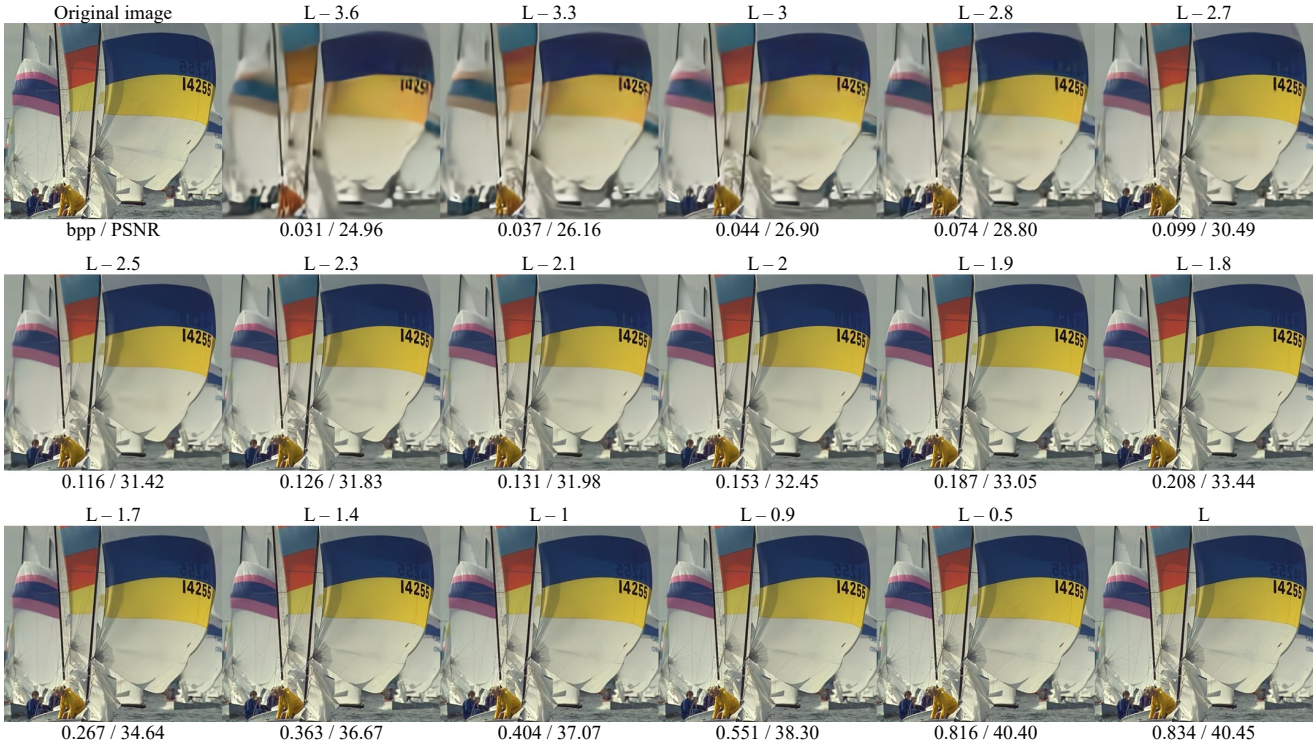
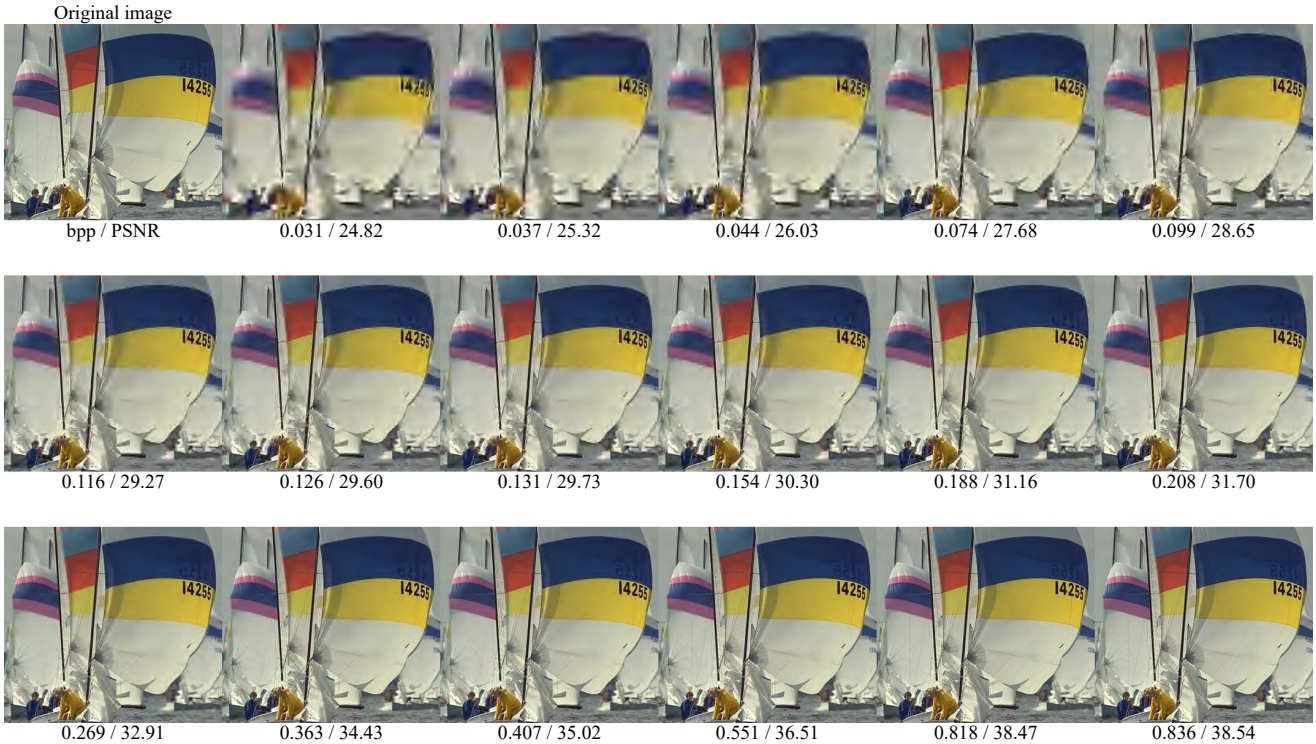


Figure S-14. Qualitative comparison of reconstructed images at similar rates in the CLIC dataset.

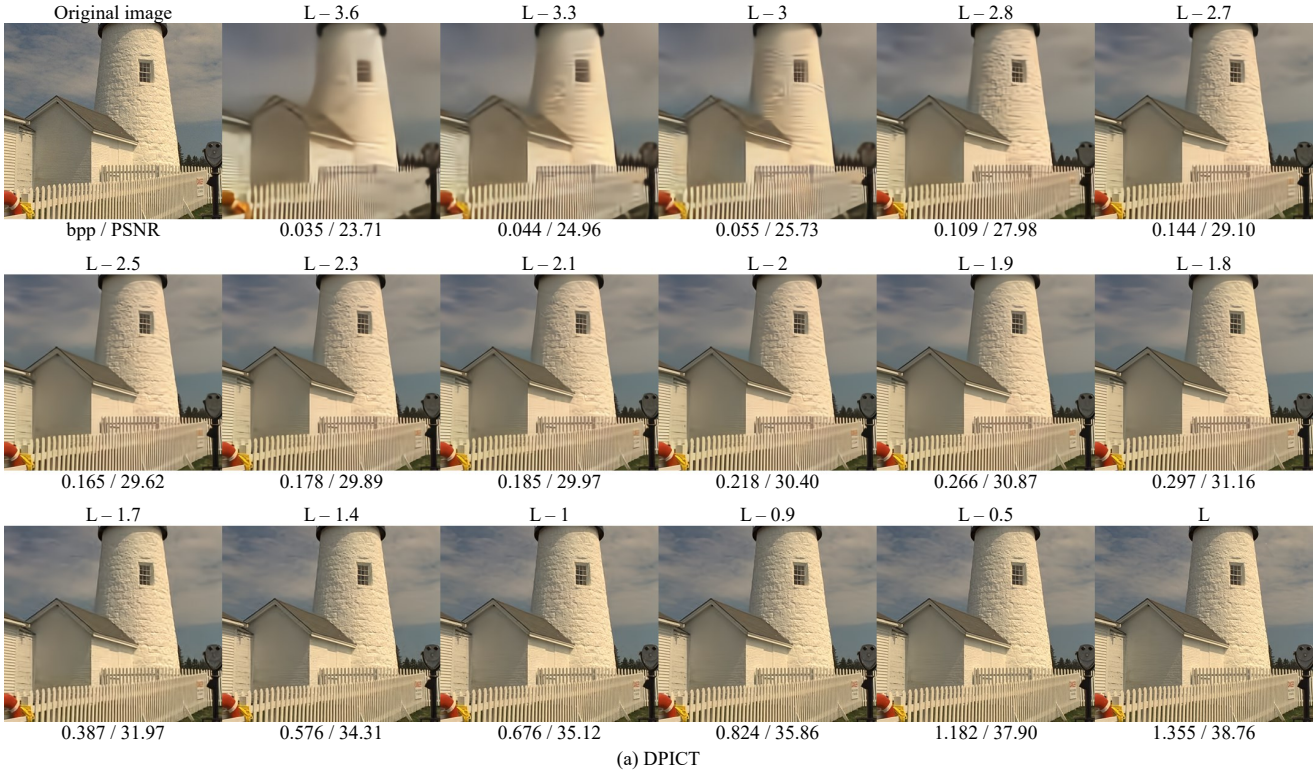


(a) DPICT



(b) JPEG2000

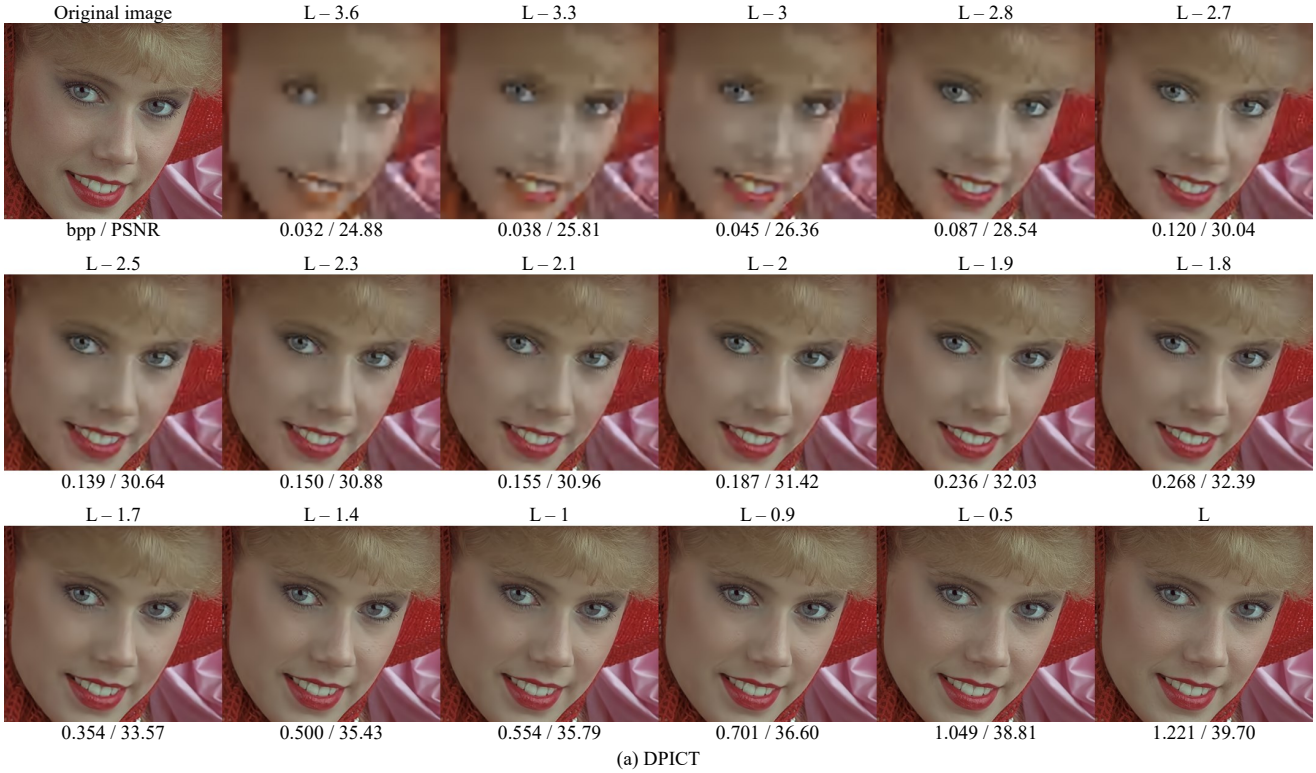
Figure S-15. Qualitative comparison of progressively reconstructed images at various rates in the Kodak dataset.



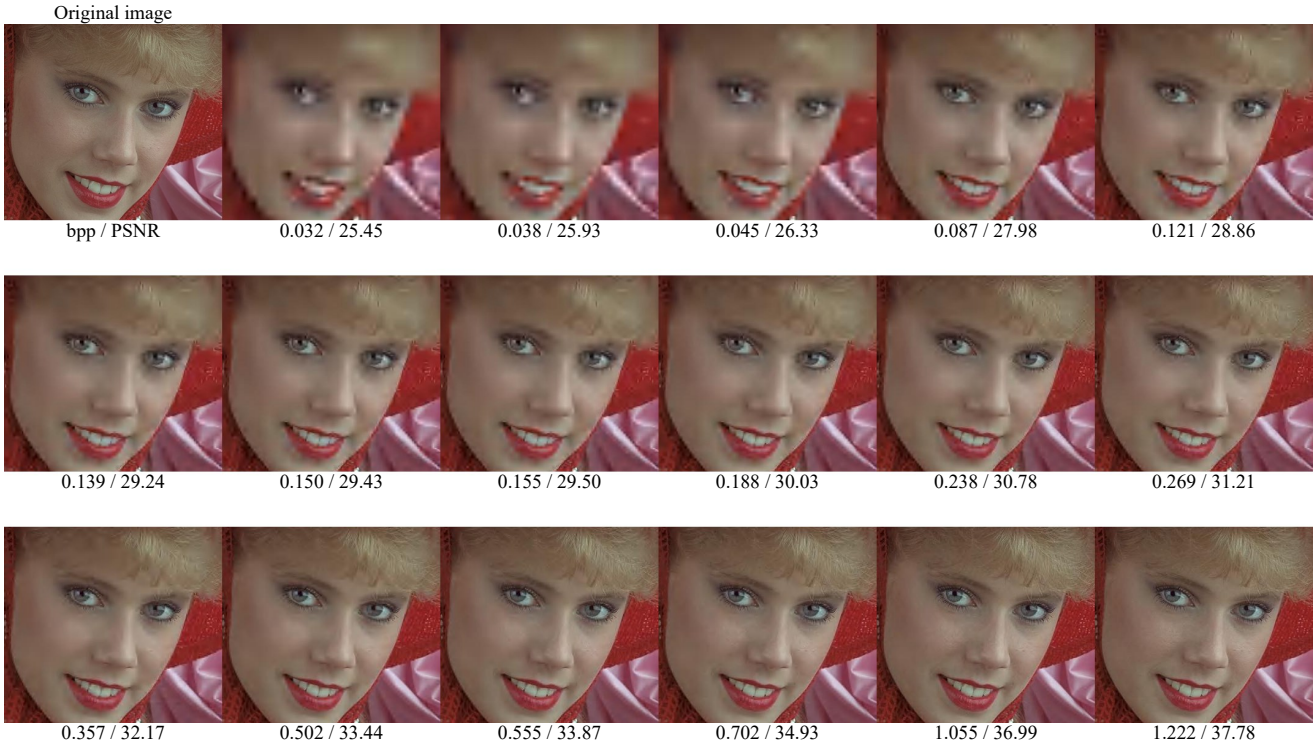
(a) DPICT

(b) JPEG2000

Figure S-16. Qualitative comparison of progressively reconstructed images at various rates in the Kodak dataset.

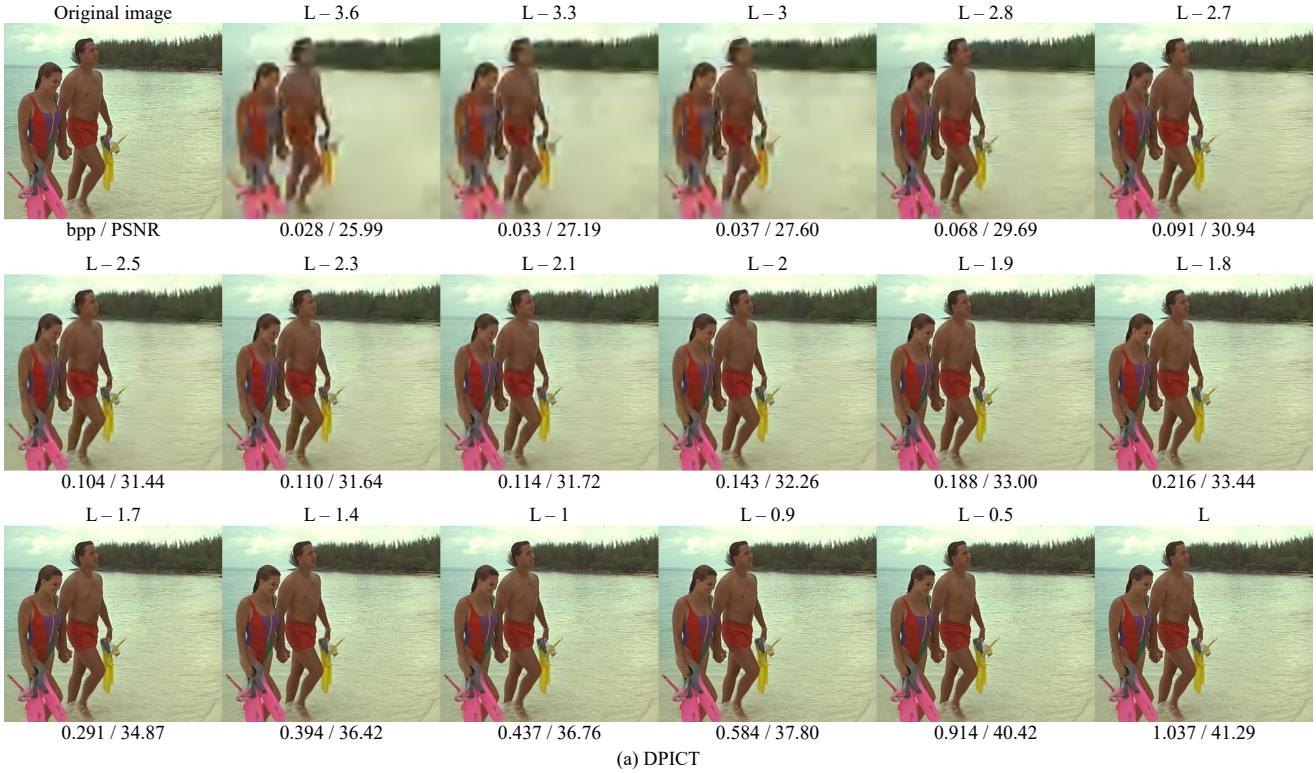


(a) DPICT

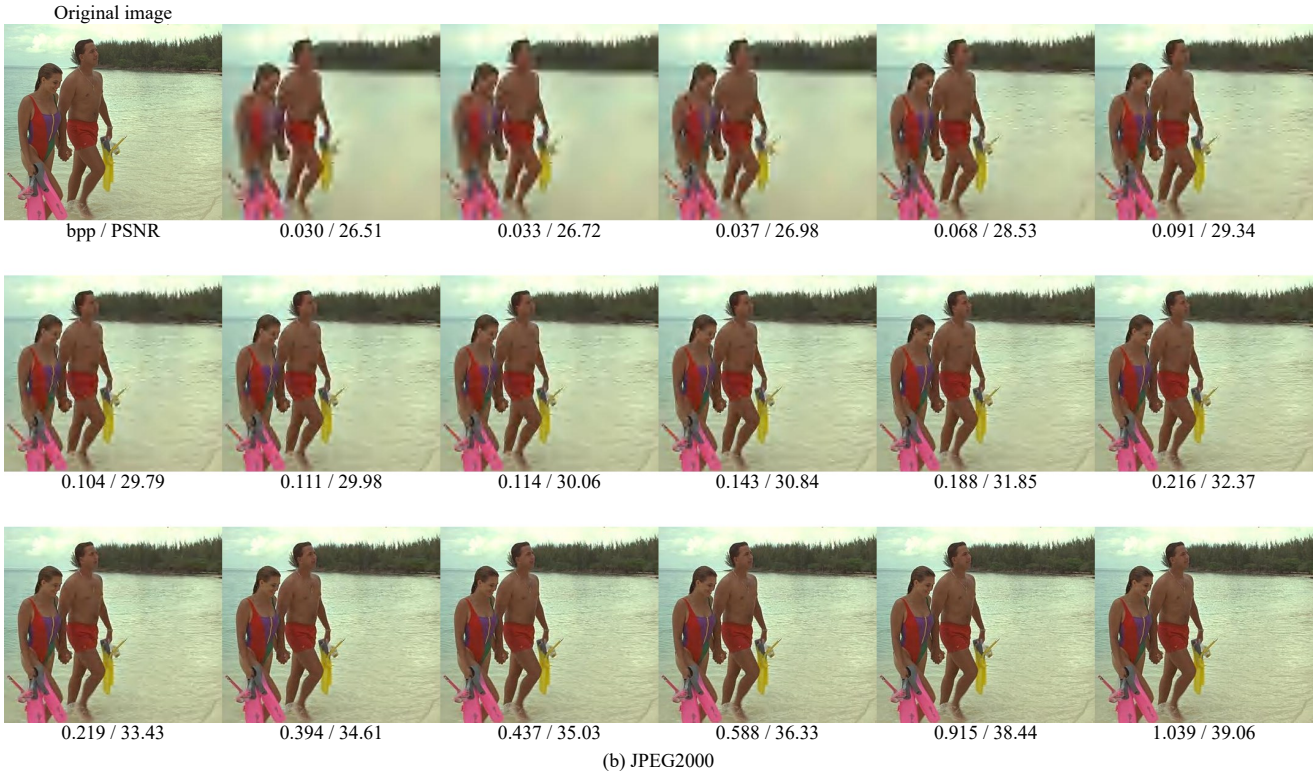


(b) JPEG2000

Figure S-17. Qualitative comparison of progressively reconstructed images at various rates in the Kodak dataset.

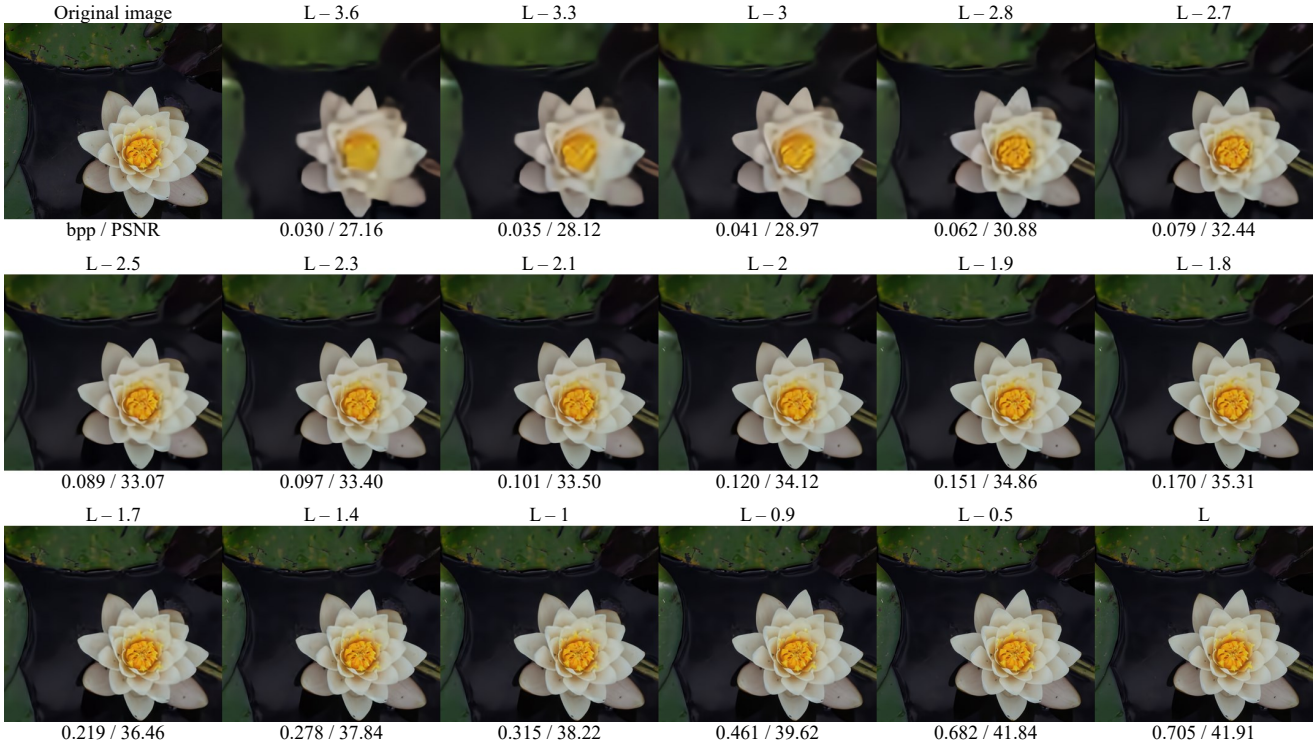


(a) DPICT

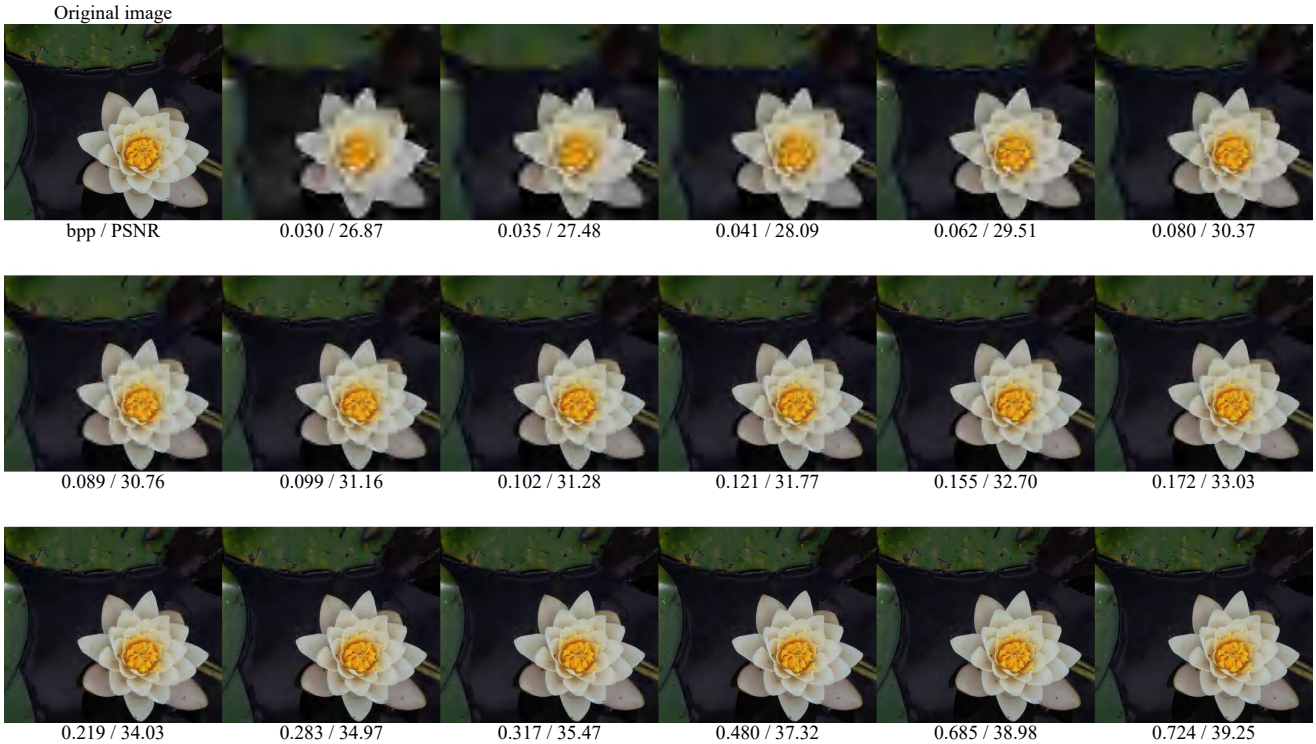


(b) JPEG2000

Figure S-18. Qualitative comparison of progressively reconstructed images at various rates in the Kodak dataset.



(a) DPICT



(b) JPEG2000

Figure S-19. Qualitative comparison of progressively reconstructed images at various rates in the CLIC dataset.

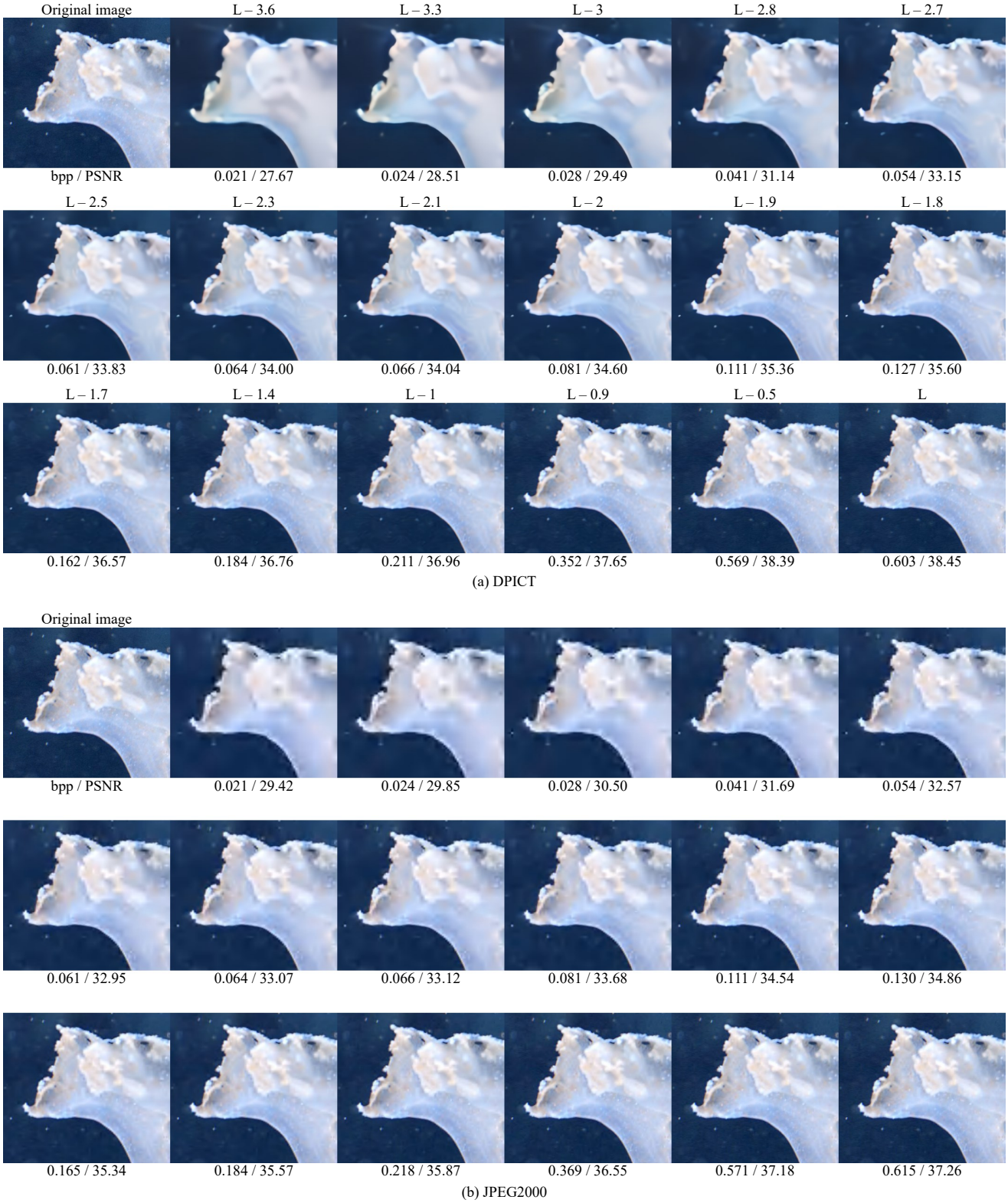
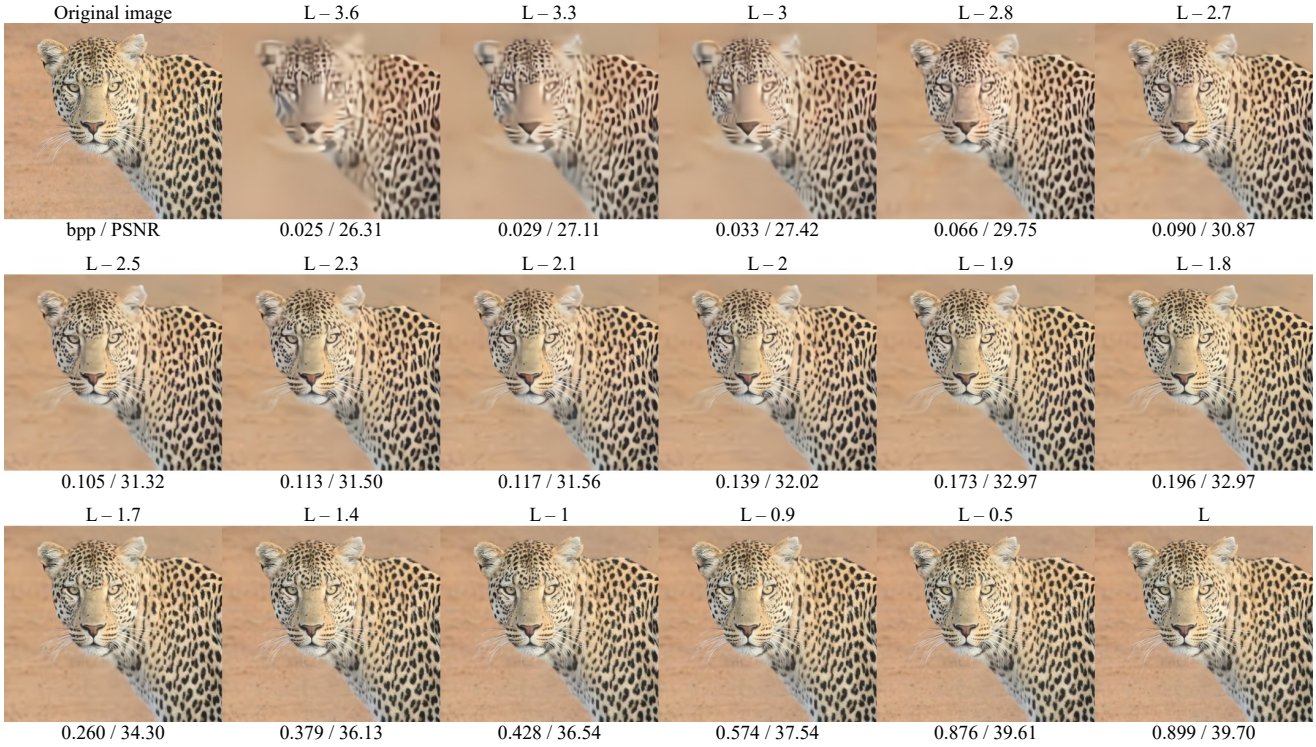
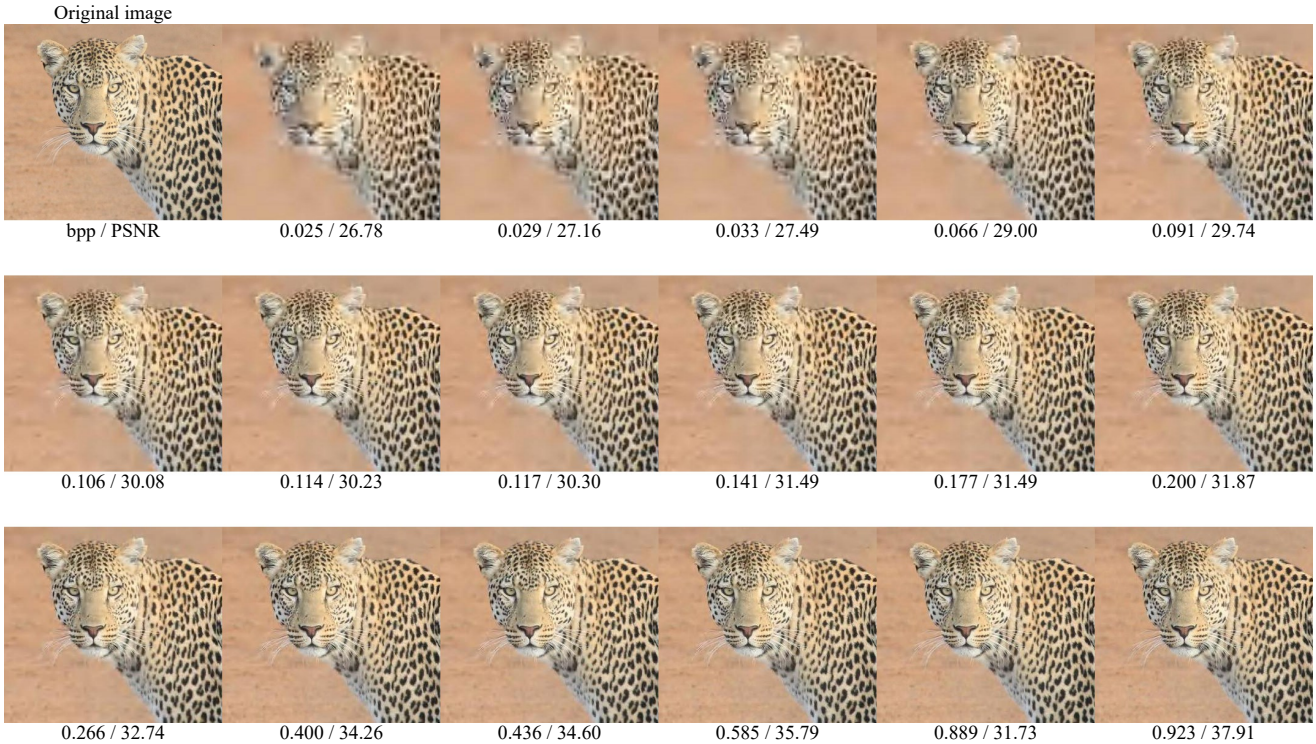


Figure S-20. Qualitative comparison of progressively reconstructed images at various rates in the CLIC dataset.



(a) DPICT



(b) JPEG2000

Figure S-21. Qualitative comparison of progressively reconstructed images at various rates in the CLIC dataset.

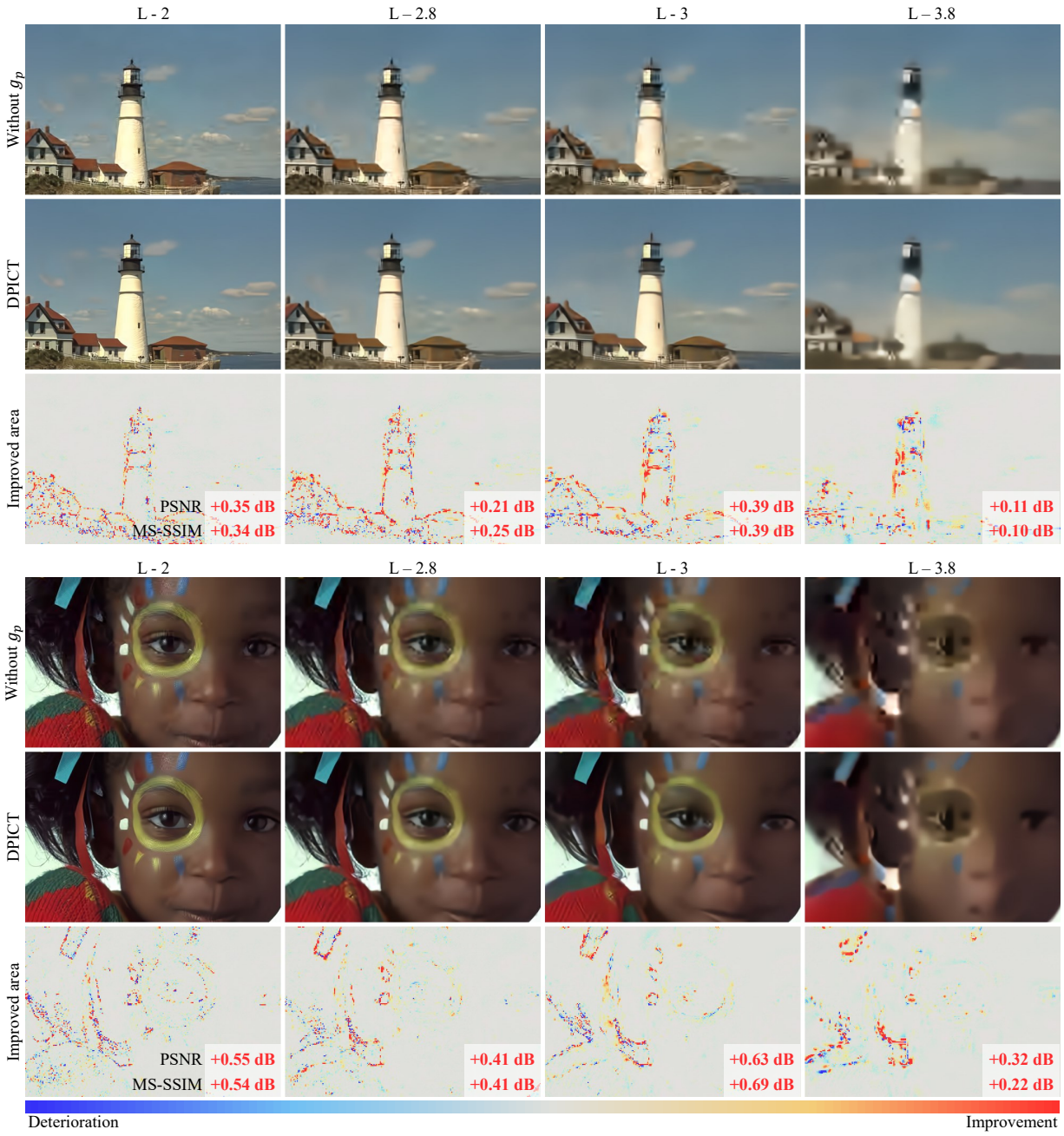


Figure S-22. Comparison of reconstructed images before and after the postprocessing in the Kodak dataset. Improvement maps are also provided.

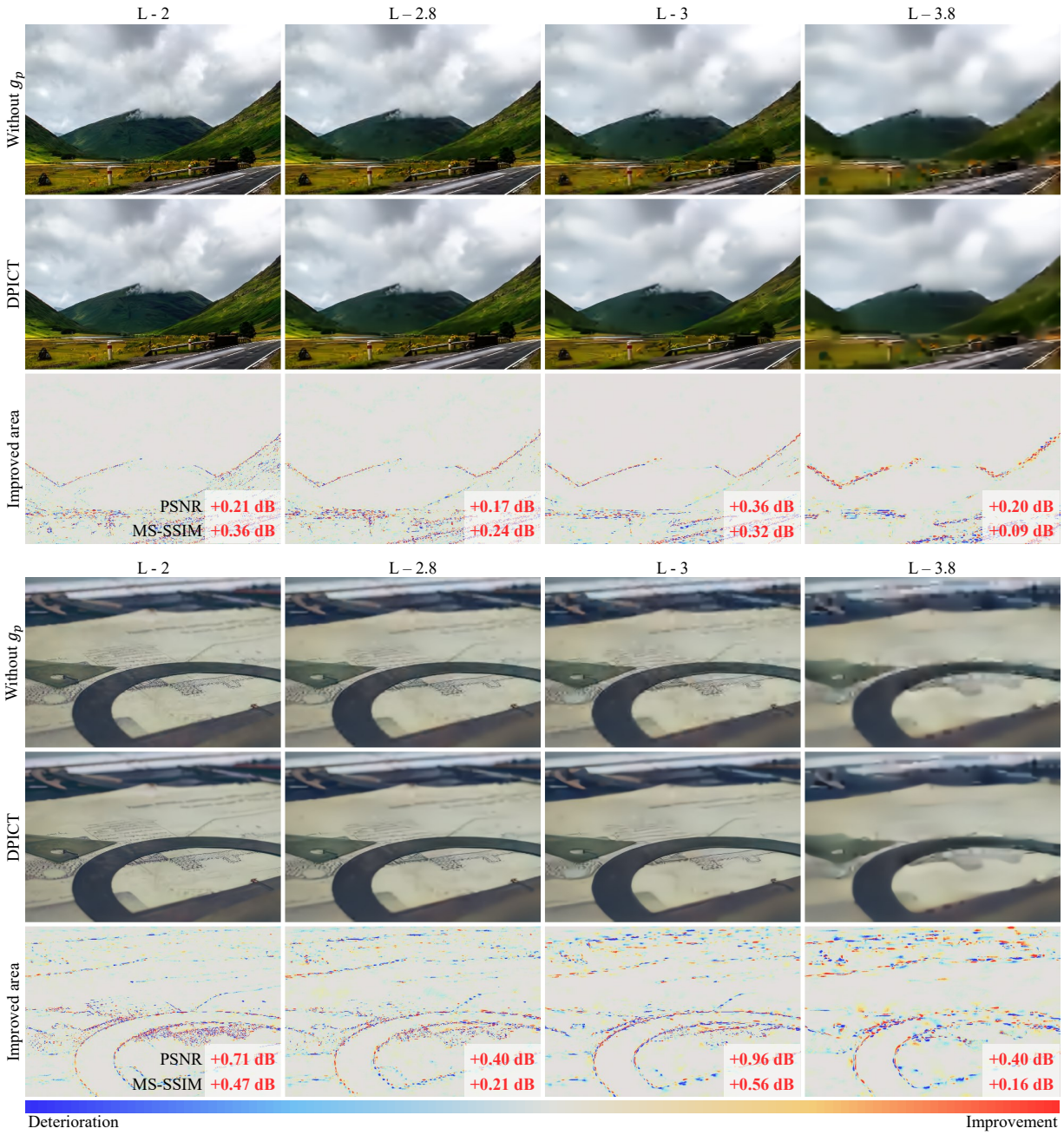


Figure S-23. Comparison of reconstructed images before and after the postprocessing in the CLIC dataset. Improvement maps are also provided.

References

[S1] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. In *ICLR*, 2017. – 1 –