ProGIC: Progressive and Lightweight Generative Image Compression with Residual Vector Quantization

Supplementary Material

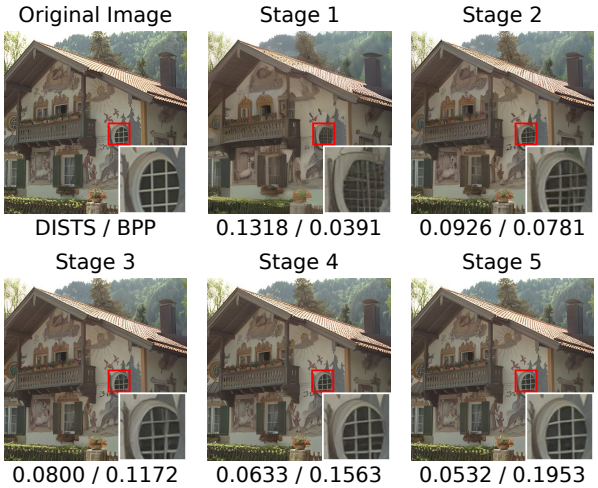


Figure 1. Visualization of progressive image transmission with ProGIC. Reconstructions are shown across increasing stages, where the cumulative BPP increases and visual fidelity improves. Values beneath each image denote DISTS / BPP.

Figure 2. Reconstruction quality comparison across different resolutions, both compressed to a total of 1280 bytes. The left column shows the original image. The middle column shows the reconstructed image at 0.04 BPP and 512×512 resolution. The right column shows the reconstructed image at 0.16 BPP and 256×256 resolution. Resolution and compression FLOPs are indicated above and below each result, respectively.

A. Progressive Decoding

Many GICs are designed for extremely low-bitrate scenarios [20, 22, 26, 27], where progressive decoding plays a critical role. Progressive decoding enables a usable image preview from only a portion of the bitstream, without waiting for the full transmission to complete, thus allowing the receiver to react faster to the image content. This requirement is especially prevalent in satellite communications [13]. A concrete usage example in a satellite communication setting is provided later in this section.

The progressive decoding behavior of ProGIC is illustrated in Fig. 1. After decoding the first stage, the main semantic content of the image is already largely recovered, which meets the demand for rapid preview. As more bits from subsequent stages arrive, the reconstructed image gradually approaches the original, and the visual discrepancy diminishes.

B. BPP Selection

Unlike GICs designed for extremely low-bitrate scenarios [20, 22, 26, 27], ProGIC targets higher bitrates. In practical transmission, latency is determined by the total bitstream length rather than by BPP. For example, a 256×256 image at 0.16 BPP and a 512×512 image at 0.04 BPP both yield approximately the same bitstream size of 1280 bytes. Fig. 2 compares the reconstruction quality under these two configurations.

We observe that at lower BPP, models tend to "generate" fine details, often producing unrealistic textures. In contrast, the lower resolution and higher BPP setting is slightly blurrier in overall appearance than the full resolution reconstruction, but it recovers image details more faithfully. Such faithful detail preservation is crucial in low-bitrate applications such as emergency response and satellite communications, where accurate decisions rely on authentic visual cues.

Moreover, using lower-resolution inputs leads to significantly fewer FLOPs, resulting in faster encoding, which is especially beneficial on edge devices. The middle image in Fig. 2 requires 222.3 GFLOPs to process, whereas the right image requires only 55.6 GFLOPs. Nevertheless, the higher BPP configuration is preferred to ensure the authenticity and reliability of reconstructed details.

C. Codebook Properties

In this section, we further analyze the properties of the codebooks in ProGIC.

C.1. Entropy of Codebook Indices

As discussed in the main text, after RVQ the entropy of the codewords is very high, which results in limited gains from entropy coding. To verify this, all RVQ indices obtained on the ImageNet dataset are collected, their empirical probability distribution is computed, and the resulting entropy is

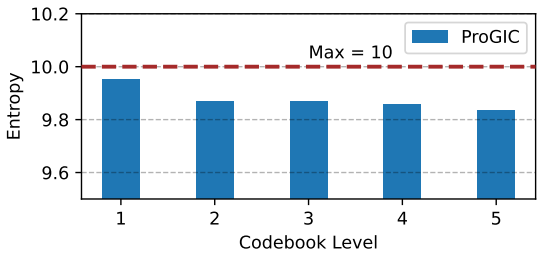


Figure 3. Entropy of different codebook usages in ProGIC.

measured, as shown in Fig. 3. Since each codebook contains $2^{10} = 1024$ codewords, the maximum possible entropy is 10, which corresponds to a uniform distribution where all indices occur with equal probability. Our results show that the entropy of every codebook exceeds 9.80, indicating that the codewords are used nearly uniformly.

C.2. Results for Entropy Coding

As shown in Fig. 4, we compare rate-distortion (R–D) curves with and without entropy coding across different datasets. We adopt range coding [17], which is consistent with most existing learned image compression methods [3, 8, 11, 15]. Since the prior probability distribution used by the entropy coder is learned from a large amount of data, there is an inevitable discrepancy between this distribution and the true probability distribution of image indices. As a result, the achieved rates are slightly above the theoretical entropy limit, which further reduces the practical gain from entropy coding. Considering all these factors, entropy coding is not used in ProGIC.

C.3. Analysis for Latent Features

Furthermore, we analyze the latent features $\boldsymbol{y}$ and the quantized residuals at each stage on the Kodak [12] dataset, and visualize them using t-SNE, as shown in Fig. 5. The original latent features $\boldsymbol{y}$ exhibit a relatively concentrated distribution. After the first-stage quantization, the quantized vectors show mild dispersion. In contrast, the residuals from subsequent stages are distributed more uniformly across the space. This suggests that early codebooks capture high-level semantic information, whereas later codebooks focus on fine-grained differences between the reconstructed and original features, leading to more uniform spatial distributions in their visualizations. This observation aligns with the idea of DLF [27], which directly models residuals using an entropy model and entropy coding. Similarly, ProGIC employs multiple additional codebooks to represent these residuals.

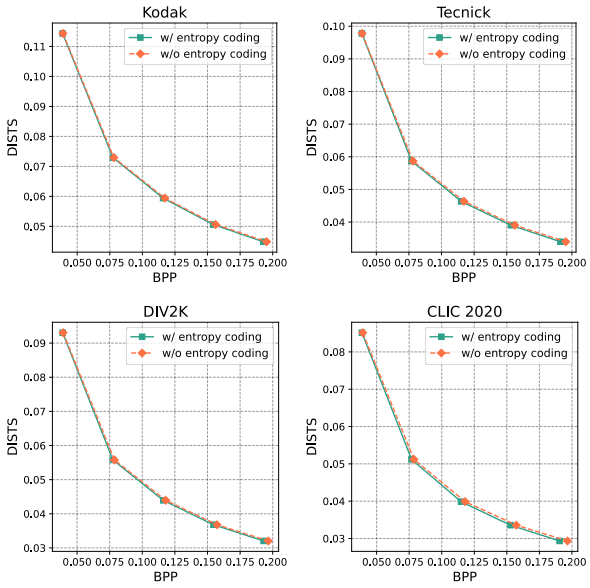


Figure 4. Rate-distortion curves with and without entropy coding.

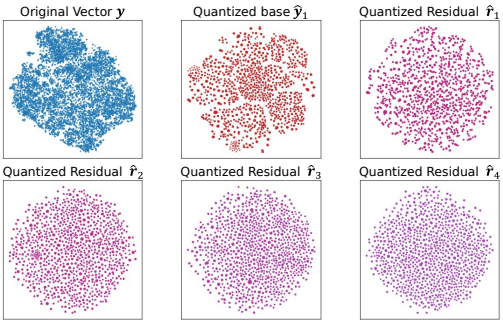


Figure 5. Visualization by t-SNE of latent features. The top-left plot shows the distribution of the unquantized latent vector $\boldsymbol{y}$. The subsequent plots illustrate the distributions of the base vector $\hat{\boldsymbol{y}}_1$ and the residuals from each quantization stage, respectively.

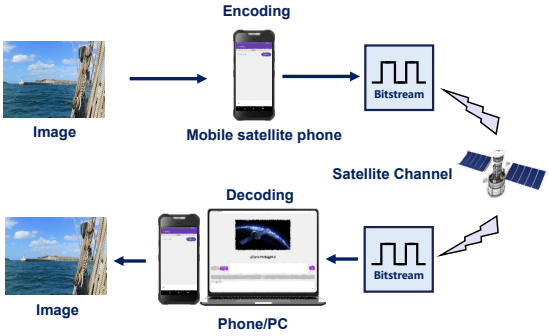
C.4. More BPP Granularities with different codebook untilizations

In the main text, results are reported for models at BPP values of $0.0391, 0.0781, 0.1172, 0.1562, 0.1953$. The achievable BPP range and step size are determined by three key parameters: the number of codebooks N, the number of codewords per codebook 2^L, and the downsampling factor f, according to

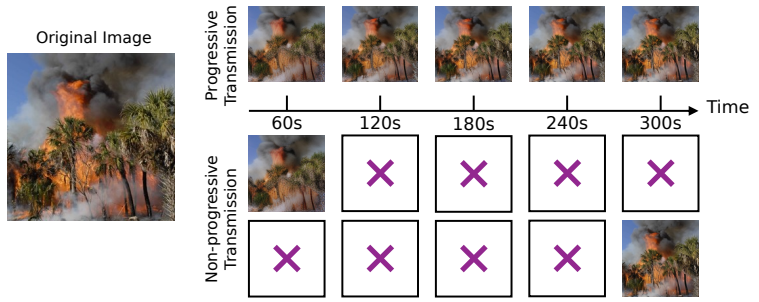
$$\text{BPP} = \frac{L}{f \times f} \times N, \tag{1}$$

where $\frac{L}{f \times f}$ represents the minimum BPP increment per progressive stage.

As shown in Fig. 7, different combinations of N and f yield different BPP ranges. The left panel corresponds to



(a) Satellite communication for mobile phone deployment.



(b) Comparison between progressive and non-progressive transmission.

Figure 6. In the event of a forest fire, ProGIC enables rapid response by transmitting images over a satellite short message link, assuming one transmission every 60 seconds. The leftmost column shows the original image. The first row illustrates progressive transmission: a usable preview is available immediately and is progressively refined as more bits arrive. In contrast, the second and third rows depict non-progressive schemes, which require the full bitstream before decoding. The second row uses a low BPP, yielding a fast but fixed-quality preview that cannot be improved later. The third row uses a high BPP, delivering higher fidelity but only after a long wait for the complete transmission, which delays situational awareness.

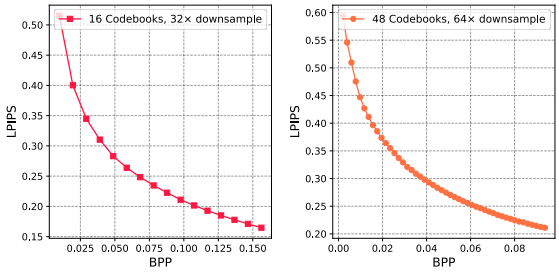


Figure 7. Different BPP ranges achieved by varying the number of codebooks and downsampling factors. The left panel shows the case of 16 codebooks with a downsampling factor of 32, with a minimal BPP increment of 0.009766. The right panel shows the case of 48 codebooks with a downsampling factor of 64, with a minimal BPP increment of 0.001953.

$N = 16$, $L = 10$, and $f = 32$, supporting 16 progressive stages with a minimal BPP increment of 0.009766. The right panel shows $N = 48$, $L = 8$, and $f = 64$, enabling 48 stages with a finer BPP increment of 0.001953. By adjusting these parameters, ProGIC can flexibly accommodate diverse transmission requirements.

D. Use Case: Mobile Phone Deployment for Satellite Communication

In this section, a practical application of ProGIC in satellite short message communication is presented—a critical solution for remote areas that lack cellular coverage, where satellite links are often the only means of connectivity. Timely disaster monitoring in such regions is vital, as early detection and rapid response can significantly mitigate losses. For instance, in the case of a forest fire, prompt identification enables immediate containment measures to prevent large-scale spread.

The satellite communication system is illustrated in Fig. 6a. Existing satellite short message systems typically support payloads of a few hundred bytes per packet, with transmission intervals ranging from 10 to 60 seconds. Assuming a payload capacity of 360 bytes per packet with a 60-second interval, ProGIC is configured with $N = 5$, $L = 10$, and $f = 16$, so that each packet carries 320 bytes of compressed data. The ProGIC model is exported to TorchScript format, packaged as an SDK, and deployed on a satellite short message handheld device powered by the MediaTek Helio P35 SoC for on-device encoding and decoding. A Java-based web interface is also developed for decoding and visualization on desktop clients. Because ProGIC does not use entropy coding, it avoids the cross-platform precision inconsistencies commonly observed in other learned image codecs [10], ensuring reliable and consistent decoding across diverse hardware platforms.

The simulation results are shown in Fig. 6b. Owing to its progressive design, ProGIC delivers an immediate coarse preview upon receiving the first packet, and the image quality progressively improves as subsequent packets arrive. In contrast, non-progressive codecs must wait for all packets to be received before decoding. If the BPP is set too low, the final reconstruction lacks clarity. If it is set too high, the waiting time becomes impractical. This demonstrates that the progressive transmission of ProGIC is particularly well suited to bandwidth- and latency-constrained satellite short message scenarios.

E. Additional Experimental Results

E.1. Performance Details

This section provides additional details regarding the baselines mentioned in the main text.

For VTM [23], its intra-frame coding mode is currently one of the strongest image compression methods available. We use the relatively recent VTM-23.10 version in our experiments to ensure a fair comparison in terms of encoding and decoding time. We build the VTM project on a Linux system and run tests with the following command:

```
EncoderApp
  -i [input.yuv]
  -c encoder_intra_vtm.cfg
  -o [output.yuv]
  -b [output.bin]
  --wdt [width]
  --hgt [height]
  -q [QP]
  --InputBitDepth=8
  -fr 1
  -f 1
  --InputChromaFormat=420
```

We use YUV420-formatted input images, as this configuration yields faster runtimes. Comparing speed under this optimized setup ensures a fair evaluation.

For Control-GIC [14], we perform an exhaustive search over all granularity combinations with a step size of 0.01 and retain the best-performing configuration. We observe that Control-GIC suffers significant performance degradation when the BPP falls below 0.15. To remain consistent with the original paper, we exclude results in the BPP < 0.15 range when computing BD-rate. Additionally, we find that the encoding and decoding complexity of Control-GIC scales quadratically with the number of image pixels, whereas other models scale linearly. At a resolution of 256×256, our measured encoding and decoding times closely match those reported in the original paper. However, at the standard Kodak resolution of 512×768, Control-GIC becomes significantly slower. On DIV2K and CLIC2020, we use the official tiling function to avoid out-of-memory errors.

For OSCAR [9], we use the author-provided pretrained models and code for evaluation. However, the official implementation does not support high-resolution image testing. To avoid introducing artificial bias, we retain the out-of-memory behavior on DIV2K and CLIC2020.

The official implementation of HiFiC [18] relies on an older version of TensorFlow and is incompatible with modern GPUs such as the NVIDIA A100 or RTX 4090. For a fair comparison, we instead use a community-provided PyTorch reimplementation with its pretrained weights. All other models use official implementations and pretrained checkpoints.

Table 1. Comparison of BD-rate on the Kodak, Tecnick, DIV2K, and CLIC datasets measured with DISTS. Best results are in **bold**. Second-best are underlined.

Method	BD-rate (DISTS)			
	Kodak	Tecnick	DIV2K	CLIC
HiFiC [18]	90.08%	99.67%	100.76%	124.45%
Control-GIC [14]	34.18%	67.12%	62.09%	110.76%
MS-ILLM [19]	0.00%	0.00%	0.00%	0.00%
DiffEIC [16]	-33.79%	23.68%	15.78%	59.91%
OSCAR [9]	-50.63%	-4.76%	–	–
ProGIC-s (Ours)	-43.87%	2.86%	-30.62%	11.86%
ProGIC (Ours)	**-57.57%**	**-20.95%**	**-44.49%**	**-13.19%**

We compute LPIPS [28] using the `lpips` Python library, normalizing inputs to $[-1, 1]$ as in the official setup and using the pretrained VGG [21] network weights, which correlate better with perceptual quality than AlexNet. DISTS [7] is computed using the `DISTS_pytorch` library with inputs normalized to $[0, 1]$, following the official configuration. FLOPs are measured using the `calflops` library in Python, where we adopt the convention that 1 FLOP $= 2$ MACs.

BD-rate (Bjøntegaard Delta rate) [4] is a widely used metric to compare the average bitrate savings of one method over another across a range of quality levels. It computes the area between two rate–distortion (R–D) curves after interpolating them using a monotonic piecewise cubic Hermite interpolating polynomial (PCHIP). A negative BD-rate indicates that the proposed method achieves the same quality at a lower bitrate compared with the baseline. We use the `bjontegaard` Python library for these calculations.

E.2. Quantitative Results on DISTS

In the main text, BD-rate results are reported based on LPIPS. In this section, we also provide BD-rate results based on DISTS, as shown in Tab. 1. ProGIC achieves the best performance across all datasets.

E.3. Runtime Analysis on High-Resolution Images

In this section, we evaluate the encoding and decoding time and GPU memory usage of various models on an NVIDIA A100 across different resolutions. The results are shown in Tab. 2. Because ProGIC completes both encoding and decoding in under 200 ms at 4K resolution, any model taking longer than 10s is marked as $>$ 10s. For memory usage, models that encounter out-of-memory errors at high resolutions are marked as $>$ 80. These results demonstrate that ProGIC achieves significant speed and memory advantages

over traditional codecs, non-generative codecs, and other GICs across all tested resolutions.

E.4. Quantitative Results for Other Metrics

In this section, we report R–D performance on the Kodak, Tecnick, DIV2K, and CLIC2020-Professional datasets, evaluated using PSNR, MS-SSIM [25], and CLIP-IQA [24], as shown in Fig. 8. ProGIC lags behind in pixel-level metrics such as PSNR because higher perceptual quality often comes at the cost of lower pixel-wise fidelity [5]. While its CLIP-IQA score is slightly inferior to that of DiffEIC, DiffEIC is over two orders of magnitude slower in decoding, making it impractical for real-world applications.

E.5. More Visualization Results

In the main text, we present visual comparisons of ProGIC against other methods on the Kodak [12] dataset. In this section, we provide additional qualitative results on the Tecnick [2], DIV2K [1], and CLIC 2020 [6] datasets. The result are shown in Figs. 9 to 14.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 126–135, 2017. 5

[2] Nicola Asuni, Andrea Giachetti, et al. Testimages: a large-scale archive for testing visual devices and basic image processing algorithms. In *STAG: Smart Tools and Applications in Computer Graphics*, pages 63–70, 2014. 5

[3] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[4] Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *ITU-T SG16, Doc. VCEG-M33*, 2001. 4

[5] Yochai Blau and Tomer Michaeli. Rethinking lossy compression: The rate-distortion-perception tradeoff. In *Proceedings of the 36th International Conference on Machine Learning*, pages 675–685. PMLR, 2019. 5

[6] CLIC. Workshop and challenge on learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 5

[7] Keyan Ding, Kede Ma, Shiqi Wang, and Eero P Simoncelli. Image quality assessment: Unifying structure and texture similarity. *IEEE transactions on pattern analysis and machine intelligence*, 44(5):2567–2581, 2020. 4

[8] Donghui Feng, Zhengxue Cheng, Shen Wang, Ronghua Wu, Hongwei Hu, Guo Lu, and Li Song. Linear attention modeling for learned image compression. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 7623–7632, 2025. 2

[9] Jinpei Guo, Yifei Ji, Zheng Chen, Kai Liu, Min Liu, Wang Rao, Wenbo Li, Yong Guo, and Yulun Zhang. Oscar: One-step diffusion codec across multiple bit-rates. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2025. 4, 6

[10] Zhaoyang Jia, Bin Li, Jiahao Li, Wenxuan Xie, Linfeng Qi, Houqiang Li, and Yan Lu. Towards practical real-time neural video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12543–12552, 2025. 3, 6

[11] Wei Jiang, Jiayu Yang, Yongqi Zhai, Feng Gao, and Ronggang Wang. Mlic++: Linear complexity multi-reference entropy modeling for learned image compression. *ACM Trans. Multimedia Comput. Commun. Appl.*, 21(5), 2025. 2

[12] Kodak Lossless True Color Image Suite. http://r0k.us/graphics/kodak/, 1993. 2, 5

[13] Oltjon Kodheli, Eva Lagunas, Nicola Maturo, Shree Krishna Sharma, Bhavani Shankar, Jesus Fabian Mendoza Montoya, Juan Carlos Merlano Duncan, Danilo Spano, Symeon Chatzinotas, Steven Kisseleff, Jorge Querol, Lei Lei, Thang X. Vu, and George Goussetis. Satellite communications in the new space era: A survey and future challenges. *IEEE Communications Surveys & Tutorials*, 23(1):70–109, 2021. 1

[14] Anqi Li, Feng Li, Yuxi Liu, Runmin Cong, Yao Zhao, and Huihui Bai. Once-for-all: Controllable generative image compression with dynamic granularity adaptation. In *International Conference on Learning Representations (ICLR)*, 2025. 4, 6

[15] Yuqi Li, Haotian Zhang, Li Li, and Dong Liu. Learned image compression with hierarchical progressive context modeling. In *The Twentieth IEEE/CVF International Conference on Computer Vision*, 2025. 2, 6

[16] Zhiyuan Li, Yanhui Zhou, Hao Wei, Chenyang Ge, and Jingwen Jiang. Towards extreme image compression with latent feature guidance and diffusion prior. *IEEE Transactions on Circuits and Systems for Video Technology*, 35(1):888–899, 2025. 4, 6

[17] G Nigel N Martin. Range encoding: an algorithm for removing redundancy from a digitised message. In *Proc. Institution of Electronic and Radio Engineers International Conference on Video and Data Recording*, 1979. 2

[18] Fabian Mentzer, George D Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. *Advances in neural information processing systems*, 33:11913–11924, 2020. 4, 6

[19] Matthew J Muckley, Alaaeldin El-Nouby, Karen Ullrich, Hervé Jégou, and Jakob Verbeek. Improving statistical fidelity for neural image compression with implicit local likelihood models. In *International Conference on Machine Learning (ICML)*, pages 25426–25443. PMLR, 2023. 4, 6

[20] Linfeng Qi, Zhaoyang Jia, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Generative latent coding for ultra-low bitrate image and video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 35(10):10500–10515, 2025. 1

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4

[22] Zhang Tianyu, Luo Xin, Li Li, and Liu Dong. Stablecodec: Taming one-step diffusion for extreme image compression.

Table 2. Comparison of GPU runtimes (ms) and memory (GB) for image encoding and decoding across different resolutions. Enc./Dec. denote encoding/decoding times. Mem. denotes memory usage. Best results are in **bold**. Second-best are underlined.

Method	512×768			1080×1920			1440×2560			2160×3840		
	Enc.	Dec.	Mem.	Enc.	Dec.	Mem.	Enc.	Dec.	Mem.	Enc.	Dec.	Mem.
VTM-23.10 [23]	>10s	150.30	–	>10s	230.10	–	>10s	288.16	–	>10s	486.71	–
LIC-HPCM [15]	62.37	82.88	0.53	309.80	342.95	1.98	465.91	474.49	2.84	1121.92	1147.79	6.05
DCVC-RT [10]	14.09	17.08	0.34	76.68	59.87	1.04	135.87	102.95	1.73	259.86	197.83	3.63
HiFiC [18]	526.51	1408.60	1.14	2894.55	6909.92	2.97	5179.44	>10s	4.78	>10s	>10s	9.75
Control-GIC [14]	103.56	436.26	6.53	610.76	2186.30	69.99	1190.97	4361.94	34.93	–	>10s	–
MS-ILLM [19]	165.38	147.79	1.12	350.85	379.01	2.99	516.47	601.18	4.87	1305.93	1613.82	9.94
DiffEIC [16]	210.18	4661.74	6.86	–	>10s	–	–	>10s	–	–	>10s	–
OSCAR [9]	53.04	167.56	5.57	513.20	1123.38	24.75	–	–	>80	–	–	>80
ProGIC (Ours)	7.64	10.99	0.66	28.40	55.00	1.07	38.63	80.38	1.46	78.26	145.24	2.58
ProGIC-s (Ours)	**6.13**	**7.66**	**0.34**	**10.32**	**17.24**	**0.66**	**15.81**	**29.07**	**0.95**	**35.22**	**63.05**	**1.79**

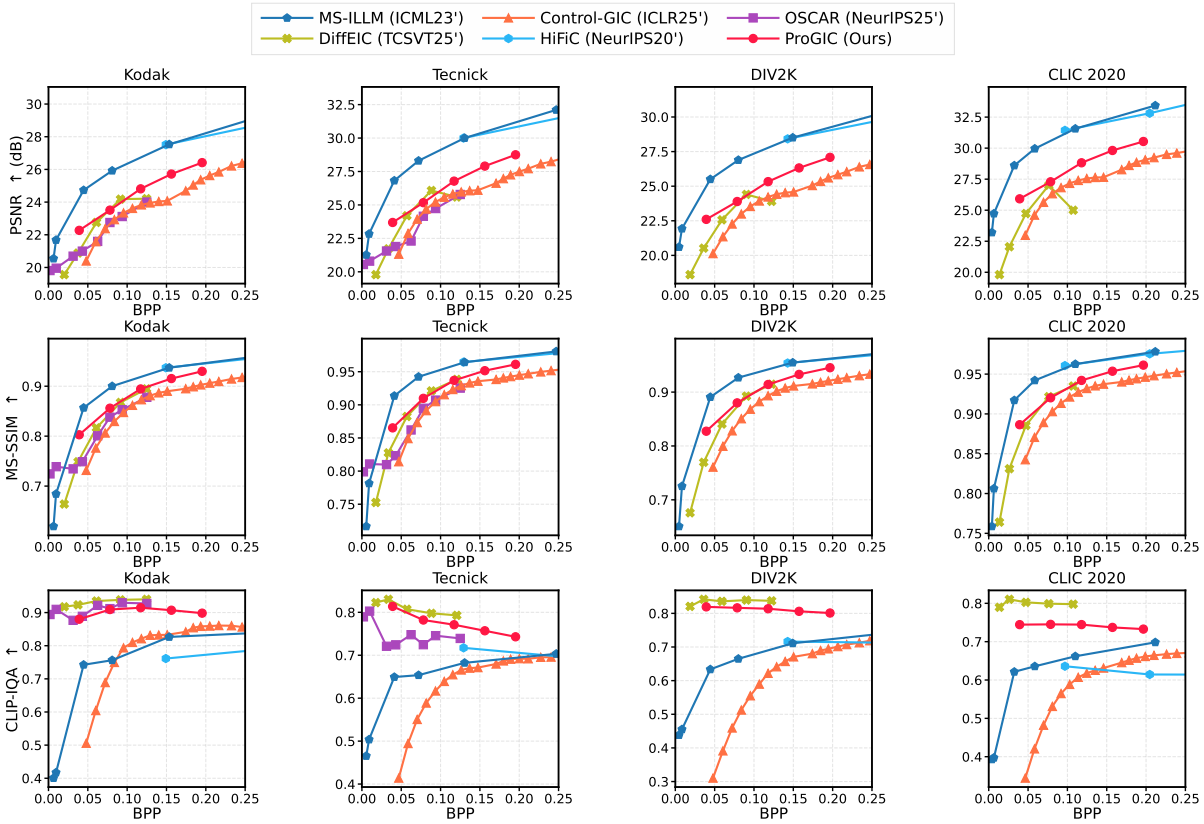


Figure 8. Rate-distortion performance on the Kodak, Tecnick, DIV2K, and CLIC2020-Professional datasets, evaluated with PSNR, MS-SSIM [25], and CLIP-IQA [24] versus BPP. Curves closer to the lower left are better, indicating higher quality at the same compression ratio.

In *International Conference on Computer Vision (ICCV)*, 2025. 1

[23] VTM-23.10. https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/, 2025. Accessed: 2025-06-05. 4, 6

[24] Jianyi Wang, Kelvin CK Chan, and Chen Change Loy. Exploring clip for assessing the look and feel of images. In *Proceedings of the AAAI conference on artificial intelligence*,

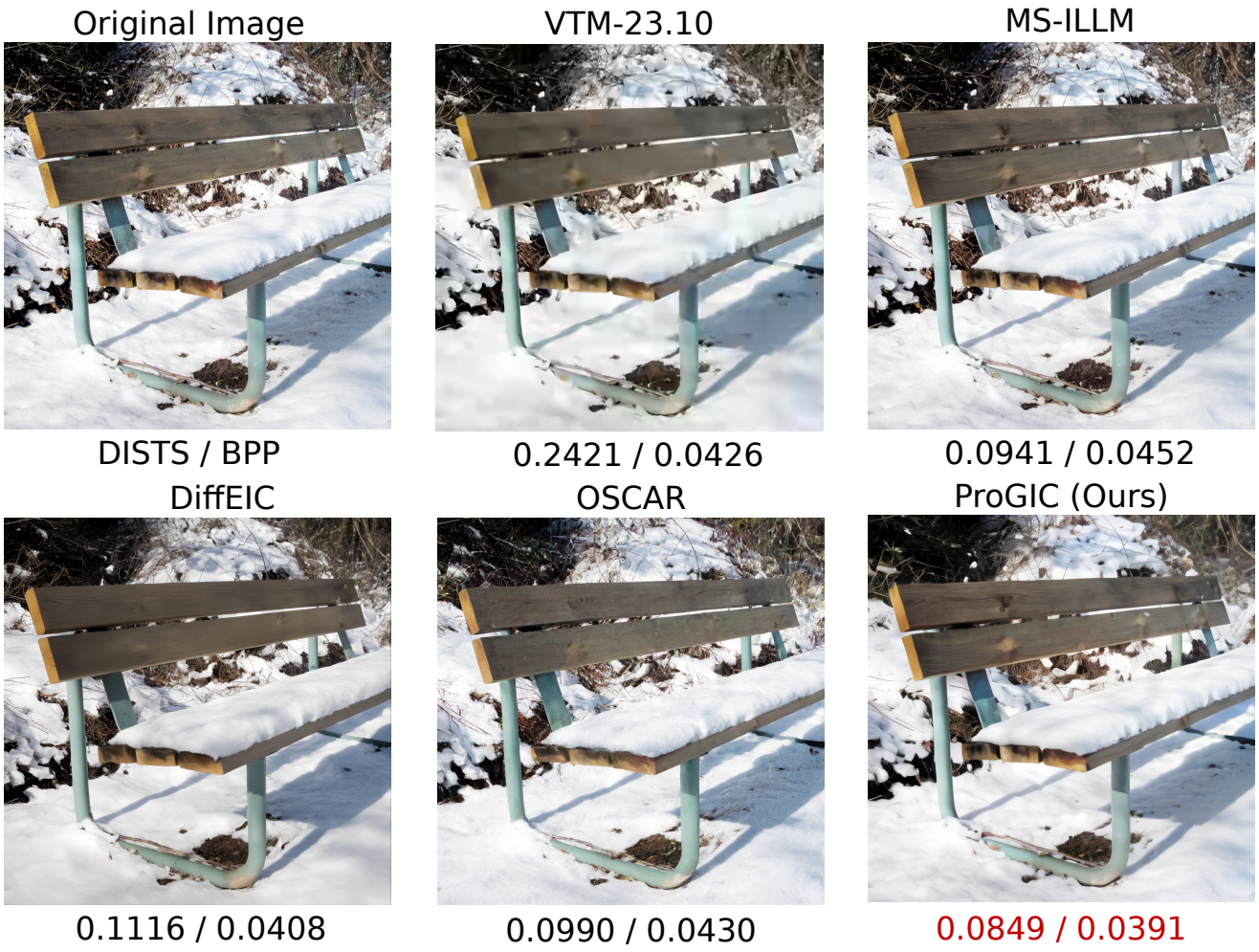


Figure 9. Visualization of reconstructed images from different methods on Tecnick. Values denote DISTS / BPP. Lower DISTS indicates better perceptual quality, and lower BPP indicates higher compression.

pages 2555–2563, 2023. 5, 6

[25] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The thirty-seventh asilomar conference on signals, systems & computers, 2003*, pages 1398–1402. IEEE, 2003. 5, 6

[26] Naifu Xue, Zhaoyang Jia, Jiahao Li, Bin Li, Yuan Zhang, and Yan Lu. One-step diffusion-based image compression with semantic distillation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2025. 1

[27] Naifu Xue, Zhaoyang Jia, Jiahao Li, Bin Li, Yuan Zhang, and Yan Lu. Dlf: Extreme image compression with dual-generative latent fusion. In *International Conference on Computer Vision (ICCV)*, 2025. 1, 2

[28] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 4

Figure 10. Visualization of reconstructed images from different methods on Tecnick. Values denote DISTS / BPP. Lower DISTS indicates better perceptual quality, and lower BPP indicates higher compression.

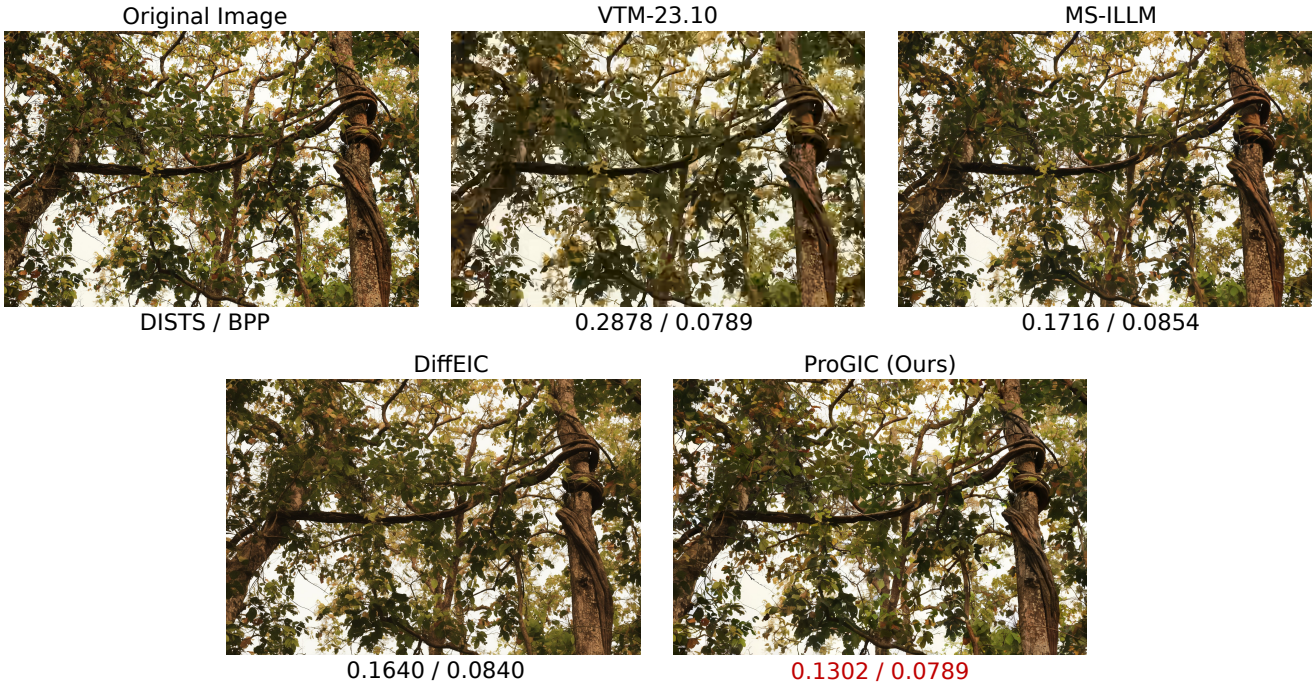


Figure 11. Visualization of reconstructed images from different methods on DIV2K. Values denote DISTS / BPP. Lower DISTS indicates better perceptual quality, and lower BPP indicates higher compression.

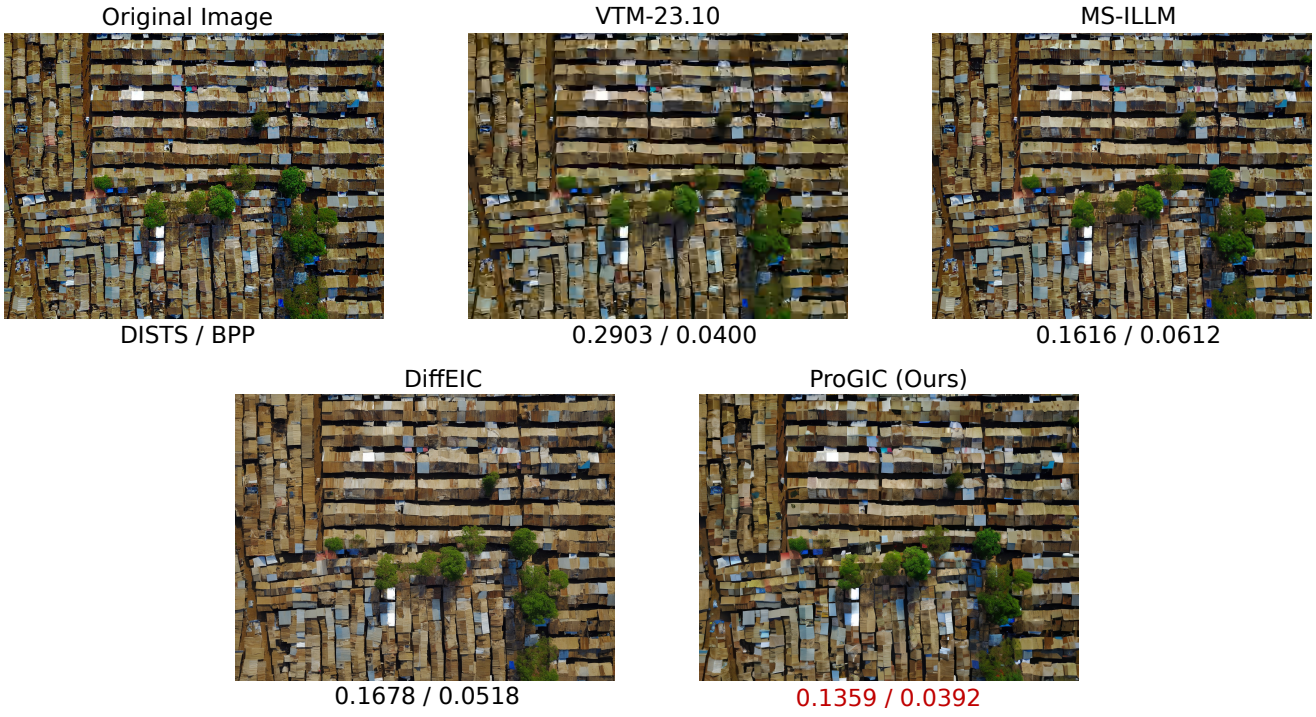


Figure 12. Visualization of reconstructed images from different methods on DIV2K. Values denote DISTS / BPP. Lower DISTS indicates better perceptual quality, and lower BPP indicates higher compression.

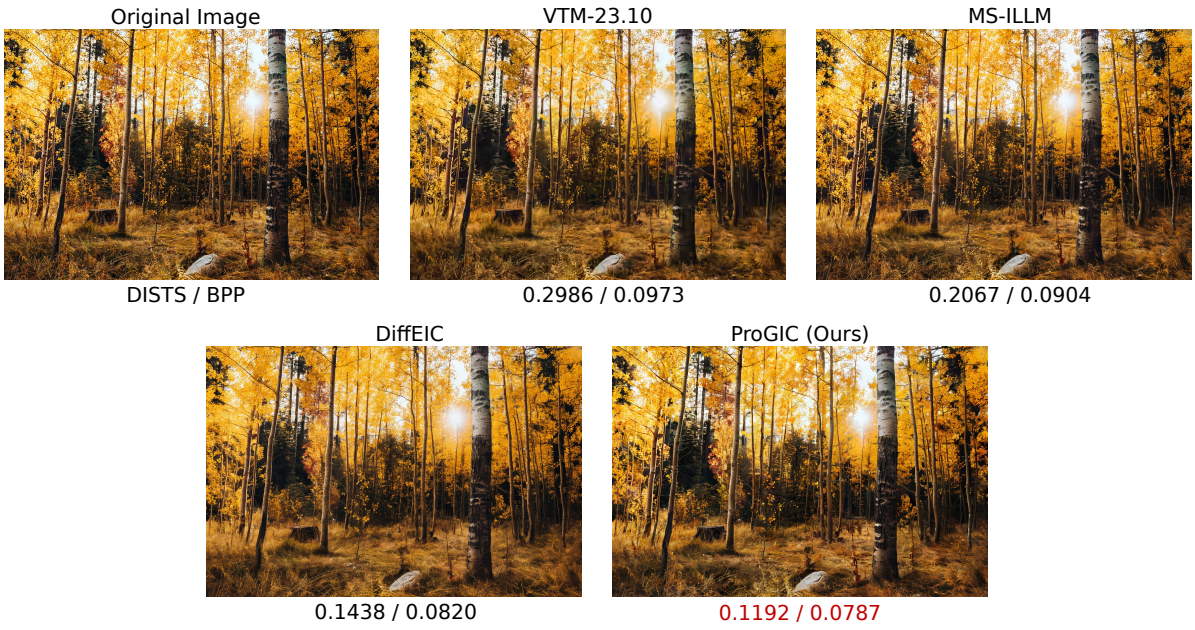


Figure 13. Visualization of reconstructed images from different methods on CLIC 2020. Values denote DISTS / BPP. Lower DISTS indicates better perceptual quality, and lower BPP indicates higher compression.

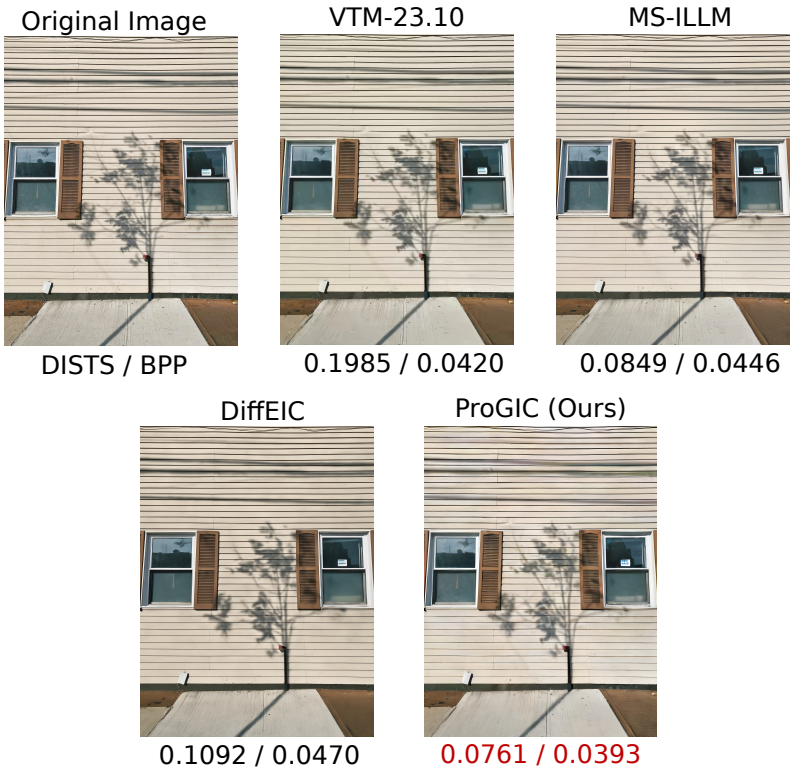


Figure 14. Visualization of reconstructed images from different methods on CLIC 2020. Values denote DISTS / BPP. Lower DISTS indicates better perceptual quality, and lower BPP indicates higher compression.