
Appendix:

OSCAR: One-Step Diffusion Codec Across Multiple Bit-rates

Jinpei Guo^{1,2}, Yifei Ji², Zheng Chen², Kai Liu²,
Min Liu¹, Wang Rao¹, Wenbo Li³, Yong Guo⁴, Yulun Zhang^{2*}
¹Carnegie Mellon University, ²Shanghai Jiao Tong University,
³Joy Future Academy, ⁴South China University of Technology

A Supplementary Experiments

A.1 Stability of Cosine Similarity During the Second Stage Fine-tuning

Since our OSCAR relies on treating $\tilde{\mathbf{z}}$ as an intermediate diffusion state, it is important that its directional consistency with the original latent \mathbf{z}_0 is maintained throughout the second-stage fine-tuning. To verify this, we examine the training dynamics of the cosine similarity between $\tilde{\mathbf{z}}$ and \mathbf{z}_0 . As shown in Fig. 1, the negative cosine similarity curves for multiple bit-rates quickly converge to high values and remain stable thereafter. This indicates that once alignment is established in the first stage, it remains robust during joint fine-tuning with the diffusion model.

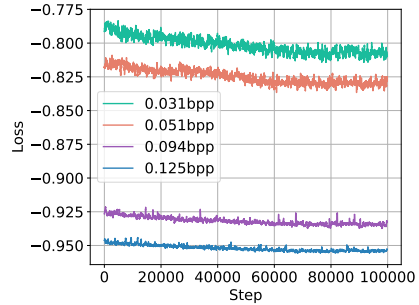


Figure 1: Training curves of negative cosine similarity between $\tilde{\mathbf{z}}$ and \mathbf{z}_0 during the second-stage fine-tuning.

Such stability is a key prerequisite for mapping bit-rates to pseudo diffusion timesteps. In particular, it ensures that the empirically measured similarity $\hat{F}_{\text{sim}}^{\phi}(\cdot)$ remains consistent and can be reliably matched to the theoretical similarity $\sqrt{\alpha_t}$. Consequently, the learned mapping $\hat{f}_{\mathcal{R} \rightarrow t}^{\phi}(\cdot)$ remains valid across training, enabling our model to support unified, one-step decoding across multiple bit-rates using a single generative backbone.

A.2 Quantitative Comparison with Multi-step Diffusion Codecs

We report quantitative results on DIV2K-val [1] and CLIC2020 [8] datasets. The baselines include three representative multi-step diffusion codecs—CDC [9], PerCo [4], and DiffeIC [6], all evaluated with 50 denoising steps. These methods are 18–35 times slower at inference than our one-step model and require a separate network for each target bit-rate.

As shown in Fig. 2, OSCAR consistently surpasses CDC and PerCo on both DIV2K-val and CLIC2020 datasets, achieving lower distortion and better perceptual quality. While DiffeIC performs slightly better at certain points, the gap remains small and within an acceptable range. These results demonstrate that our one-step model achieves performance on par with multi-step diffusion codecs, and even surpasses them in certain cases, while being significantly faster and more efficient.

*Corresponding author: Yulun Zhang, yulun100@gmail.com

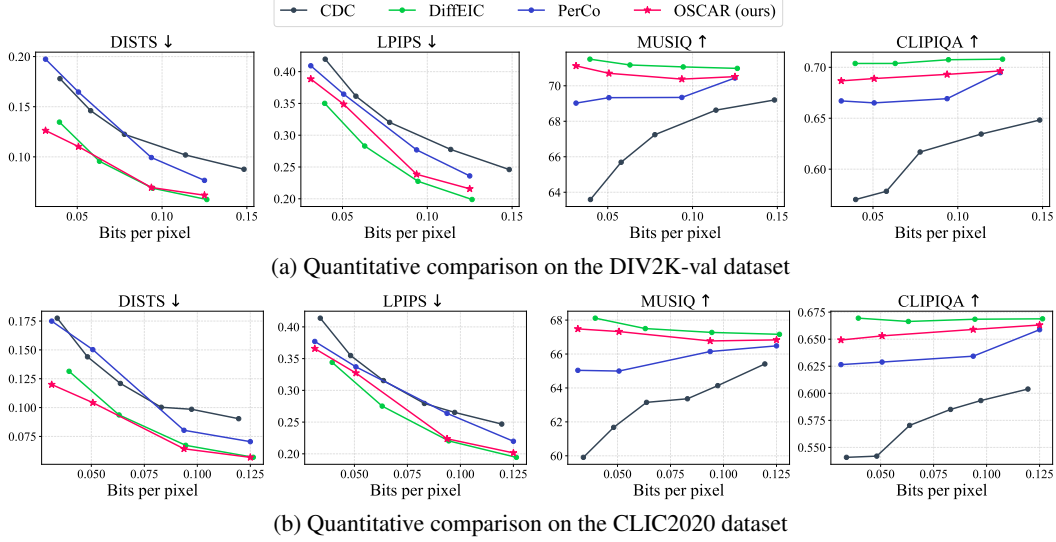


Figure 2: Evaluation of OSCAR and multi-step diffusion-based codecs on DIV2K-Val and CLIC2020.

A.3 Pixel-wise Distortion Results

To assess pixel-wise distortion performance, we report MS-SSIM results on Kodak, DIV2K-val, and CLIC2020 datasets. The baselines include the traditional codec BPG [2], the learning-based method MS-ILLM [7], and multi-step diffusion codecs CDC [9], PerCo [4], and DiffEIC [6].

As shown in Fig. 3, diffusion-based codecs generally underperform on the MS-SSIM metric primarily due to the limitations of the stable diffusion autoencoder, which prioritizes perceptual quality over pixel-wise accuracy. Nevertheless, OSCAR achieves competitive results, surpassing PerCo and CDC, and showing only a small gap compared to DiffEIC, which is 25 times slower than OSCAR, and requires training a separate model for each bit-rate.

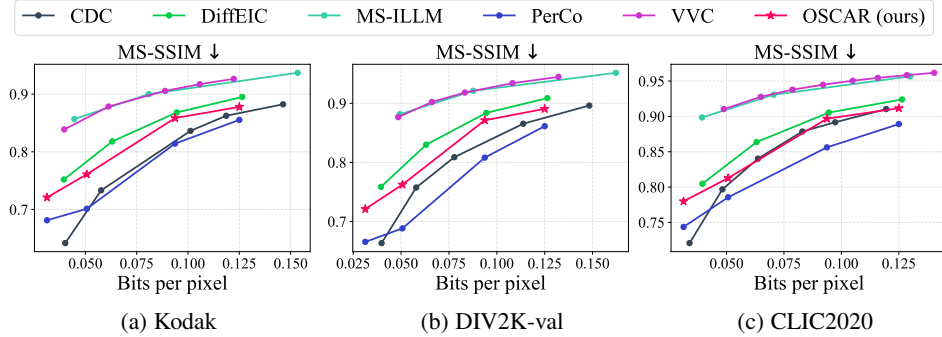


Figure 3: Evaluation of OSCAR and representative baselines on the MS-SSIM metric.

B Additional Method Details

B.1 Hyper-Encoder Architecture Details

Frontend module. Given an input latent feature $\mathbf{z}_0 \in \mathbb{R}^{N \times d}$, where N is the number of spatial tokens and d is the latent channel dimension, the frontend module hierarchically downsamples the feature map to extract multi-scale representations.

The pipeline begins with a 3×3 convolution to embed the input into feature space:

$$\mathbf{h}_0 = \text{Conv}(\mathbf{z}_0). \quad (1)$$

This is followed by a residual block (ResB) and a self-attention layer (Attn). Each ResB block adopts a bottleneck structure:

$$\text{ResB}(\mathbf{x}) = \mathbf{x} + \text{Conv}_{1 \times 1} \circ \text{ReLU} \circ \text{Conv}_{3 \times 3} \circ \text{ReLU} \circ \text{Conv}_{1 \times 1}(\mathbf{x}), \quad (2)$$

where \circ denotes function composition and each convolution is followed by a ReLU activation (except the last). The channel width is temporarily reduced to half in the bottleneck.

Next, a stride-2 convolution (Down) reduces the spatial resolution by a factor of 2:

$$\mathbf{h}_3 = \text{Down}(\mathbf{h}_2) = \text{Conv}_{\text{stride}=2}(\mathbf{h}_2). \quad (3)$$

This is followed by another residual block and attention layer. The process repeats once more:

$$\begin{aligned} \mathbf{h}_1 &= \text{ResB}(\mathbf{h}_0), \quad \mathbf{h}_2 = \text{Attn}(\mathbf{h}_1), \quad \mathbf{h}_3 = \text{Down}(\mathbf{h}_2), \quad \mathbf{h}_4 = \text{ResB}(\mathbf{h}_3), \\ \mathbf{h}_5 &= \text{Attn}(\mathbf{h}_4), \quad \mathbf{h}_6 = \text{Down}(\mathbf{h}_5), \quad \mathbf{h}_7 = \text{Attn}(\mathbf{h}_6). \end{aligned} \quad (4)$$

Each Attn module applies self-attention across tokens, enabling global contextual aggregation. Together, this encoder progressively reduces spatial resolution while enriching the feature representation at each level, producing a compact and informative encoding suited for downstream tasks.

Codebook quantization. Following the frontend module, we apply vector quantization to discretize the continuous latent representations. Specifically, we use a learnable codebook $\mathcal{E} = \{\mathbf{e}_k\}_{k=1}^K$, where each code vector $\mathbf{e}_k \in \mathbb{R}^M$ has a fixed embedding dimension of $M = 32$. The codebook size K is determined by the bit-rate. This setup follows the standard VQ-VAE quantization scheme.

Given an input feature $\mathbf{h}_7 \in \mathbb{R}^{N' \times M}$, the quantized output $\mathbf{z}_q \in \mathbb{R}^{N' \times M}$ is obtained by replacing each token with the nearest codebook vector in terms of Euclidean distance:

$$\mathbf{z}_q[i] = \arg \min_{\mathbf{e}_k \in \mathcal{E}} \|\mathbf{h}_7[i] - \mathbf{e}_k\|_2^2, \quad \forall i \in \{1, \dots, N'\}. \quad (5)$$

Backend module. The backend module reconstructs the final latent output $\tilde{\mathbf{z}} \in \mathbb{R}^{N \times M}$ from the quantized codebook features $\mathbf{z}_q \in \mathbb{R}^{N' \times M}$ via a learnable upsampling network.

We first pass \mathbf{z}_q through a stack of transposed convolutions and residual blocks to progressively upsample the spatial resolution. The overall structure mirrors the frontend in reverse:

$$\begin{aligned} \mathbf{u}_0 &= \text{Conv}(\mathbf{z}_q), \quad \mathbf{u}_1 = \text{TConv}(\mathbf{u}_0), \quad \mathbf{u}_2 = \text{ResB}(\mathbf{u}_1), \\ \mathbf{u}_3 &= \text{TConv}(\mathbf{u}_2), \quad \mathbf{u}_4 = \text{ResB}(\mathbf{u}_3), \quad \hat{\mathbf{z}} = \mathbf{u}_4. \end{aligned} \quad (6)$$

Each TConv denotes a transposed convolution (a.k.a. deconvolution) with stride 2.

To ensure consistency with the original latent \mathbf{z}_0 , we additionally apply a rescale operation that matches the norm of $\tilde{\mathbf{z}}$ to that of \mathbf{z}_0 . Specifically, we compute:

$$\tilde{\mathbf{z}} = \frac{\|\mathbf{z}_0\|}{\|\hat{\mathbf{z}}\|} \cdot \hat{\mathbf{z}}, \quad (7)$$

where both norms are computed over all elements (e.g., using ℓ_2 norm across the flattened tensor). This rescaling helps recover the correct magnitude that was discarded during quantization and facilitates accurate one-step reconstruction.

To minimize storage overhead, we store only the global norm of each channel vector in the latent representation. Specifically, for the latent space used in SD 2.1, where the channel dimension is $d = 4$, we treat $\mathbf{z}_0 \in \mathbb{R}^{N \times 4}$ as four N -dimensional vectors and store the ℓ_2 norm of each channel separately. Each norm is stored using 16 bits, resulting in a total of $4 \times 16 = 64$ bits per image. This compact representation enables accurate magnitude recovery while introducing negligible overhead.

B.2 Pseudocode for OSCAR Training and Inference

In this section, we present detailed pseudocode for OSCAR’s training and inference procedures, as illustrated in Algorithm 1 and Algorithm 2, respectively.

B.3 Complexity Analysis

Table 1 provides a comparison of model complexity, considering key factors such as multiply-accumulate operations (MACs), parameter count (Params), and the total time for compression and decompression. To ensure a fair assessment, all models are evaluated on a single NVIDIA A6000 GPU with an input image resolution of $1,024 \times 1,024$. Notably, OSCAR achieves significantly lower computational cost compared to multi-step diffusion-based codecs (CDC [9], PerCo [4], and Dif-fEIC [6]) and learning-based multi-rate methods (CTC [5]). Despite being diffusion-based, OSCAR achieves an inference speed on the same order of magnitude as traditional methods (BPG [2] and VVC [3]) and learning-based single-rate codecs (MS-ILLM [7]), demonstrating strong efficiency.

Algorithm 1 OSCAR Training Procedure

```
1: Input: Image dataset  $\mathcal{D}$ , predefined bit-rate set  $\mathcal{R}$ 
2: Initialize: Hyper-encoders  $\{Q_\varphi^r\}_{r \in \mathcal{R}}$ , diffusion U-Net  $DN_\theta$ 
3: Freeze: Stable Diffusion VAE encoder  $E_\theta$  and decoder  $D_\theta$ 
4: Stage 1: Quantized Representation Alignment
5: for  $(I, r) \sim (\mathcal{D}, \mathcal{R})$  do
6:    $z_0 \leftarrow E_\theta(I)$  ▷ Encode image to latent
7:    $\tilde{z} \leftarrow Q_\varphi^r(z_0)$  ▷ Quantize latent via bitrate-specific hyper-encoder
8:   Compute cosine similarity loss  $\mathcal{L}_{\text{sim}}^r$  and VQ loss  $\mathcal{L}_{\text{VQ}}$ 
9:   Update  $Q_\varphi^r$  to minimize  $\mathcal{L}_{\text{repa}} = \mathcal{L}_{\text{sim}}^r + \mathcal{L}_{\text{VQ}}$ 
10: end for
11: Estimate pseudo timestep mapping  $f_\varphi^{R \rightarrow t}(r)$  via cosine similarity matching
12: Stage 2: Unified Reconstruction Training
13: for  $(I, r) \sim (\mathcal{D}, \mathcal{R})$  do
14:    $z_0 \leftarrow E_\theta(I)$ 
15:    $\tilde{z} \leftarrow Q_\varphi^r(z_0)$ 
16:    $t \leftarrow f_\varphi^{R \rightarrow t}(r)$  ▷ Map bit-rate to pseudo timestep
17:    $\hat{z}_0 \leftarrow DN_\theta(\tilde{z}, t)$  ▷ Denoise quantized latent
18:    $\hat{I} \leftarrow D_\theta(\hat{z}_0)$ 
19:   Compute total loss:  $\mathcal{L} = \mathcal{L}_{\text{repa}} + \lambda_1 \mathcal{L}_{\text{per}} + \lambda_2 \mathcal{L}_G$ 
20:   Update  $\theta, \varphi$  using AdamW optimizer
21: end for
```

Algorithm 2 OSCAR Inference Procedure

```
1: Input: Test image  $I$ , desired bit-rate  $r$ 
2: Compression Phase:
3:  $z_0 \leftarrow E_\theta(I)$  ▷ Encode image using frozen VAE encoder
4:  $h \leftarrow \text{Frontend}(z_0)$  ▷ Downsampling with ResBlock + Attn + Conv
5:  $z_q, \text{indices} \leftarrow \text{VQ}(h)$  ▷ Quantize via bitrate-specific codebook
6: Save:
   • Codebook indices of  $z_q$  (discrete latent representation)
   • Channel-wise norms  $\{\|z_0^{(c)}\|\}_{c=1}^4$  (16 bits per channel)
7: Decompression Phase:
8:  $z_q \leftarrow \text{Dequantize}(\text{indices})$ 
9:  $\hat{z} \leftarrow \text{Backend}(z_q)$  ▷ Upsampling with TConv + ResBlock
10:  $\tilde{z} \leftarrow \frac{\|z_0\|}{\|\hat{z}\|} \cdot \hat{z}$  ▷ Norm rescaling
11: One-Step Reconstruction:
12:  $t \leftarrow f_\varphi^{R \rightarrow t}(r)$  ▷ Map bitrate to pseudo timestep
13:  $\hat{z}_0 \leftarrow DN_\theta(\tilde{z}, t)$  ▷ One-step latent denoising
14:  $\hat{I} \leftarrow D_\theta(\hat{z}_0)$  ▷ Decode using frozen VAE decoder
15: Return reconstructed image  $\hat{I}$ 
```

Furthermore, compared to PerCo and DiffEIC, which also build upon the foundation model SD 2.1, OSCAR uses significantly fewer parameters. This is primarily attributed to our lightweight hyper-encoder design and the omission of additional modules for extracting redundant textual information. These design choices not only reduce the model size but also simplify the overall architecture, enabling faster and more efficient inference without compromising performance.

C Proofs

Proposition 1. Let $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. Then the expectation of its Euclidean norm is

$$\mathbb{E}[\|\mathbf{z}_0\|] = \sqrt{2} \frac{\Gamma(\frac{d+1}{2})}{\Gamma(\frac{d}{2})} \approx \sqrt{d}, \quad (8)$$

Table 1: Complexity comparison among representative baselines. MACs denotes the number of multiply–accumulate operations. Inference time denotes the total time for both compression and decompression. All models are tested with an input image size of $1,024 \times 1,024$.

Method	MACs	Params	Inference Time (ms)
MS-ILLM	1.05×10^{12}	1.81×10^8	568.69
PerCo (s=50)	6.26×10^{13}	4.98×10^9	15682.44
DiffEIC (s=50)	3.05×10^{14}	2.44×10^9	21815.99
CDC (s=50)	1.89×10^{14}	5.36×10^7	30731.12
BPG	–	–	1023.69
VVC	–	–	445.39
CTC	5.56×10^{12}	3.57×10^8	13031.54
OSCAR (s=1)	8.54×10^{12}	9.87×10^8	861.74

proof. We denote $R = \|z_0\|$. Since $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, R follows the chi-distribution with d degrees of freedom, whose density is

$$f_R(r) = \frac{1}{2^{\frac{d}{2}-1} \Gamma(\frac{d}{2})} r^{d-1} e^{-\frac{r^2}{2}}, \quad r > 0. \quad (9)$$

Hence, the expectation of R can be computed as:

$$\mathbb{E}[R] = \int_0^\infty r f_R(r) dr = \frac{1}{2^{\frac{d}{2}-1} \Gamma(\frac{d}{2})} \int_0^\infty r^d e^{-\frac{r^2}{2}} dr. \quad (10)$$

With the substitution $u = r^2/2$ ($dr = \frac{1}{\sqrt{2u}} du$),

$$\mathbb{E}[R] = \frac{1}{2^{\frac{d}{2}-1} \Gamma(\frac{d}{2})} \int_0^\infty (2u)^{\frac{d}{2}} e^{-u} \frac{du}{\sqrt{2u}} = \frac{2^{\frac{1}{2}} \Gamma(\frac{d+1}{2})}{\Gamma(\frac{d}{2})}. \quad (11)$$

Apply Stirling’s formula $\Gamma(x + \frac{1}{2}) = \Gamma(x) \sqrt{x} (1 - \frac{1}{8x} + O(x^{-2}))$ with $x = \frac{d}{2}$:

$$\frac{\Gamma(\frac{d+1}{2})}{\Gamma(\frac{d}{2})} = \sqrt{\frac{d}{2}} \left(1 - \frac{1}{4d} + O(d^{-2})\right). \quad (12)$$

Multiplying by $\sqrt{2}$ yields the asymptotic expansion

$$\mathbb{E}[\|z_0\|] = \sqrt{d} \left(1 - \frac{1}{4d} + O(d^{-2})\right) \approx \sqrt{d}. \quad (13)$$

Proposition 2. *In the forward diffusion process, the expected cosine similarity between the intermediate state \mathbf{z}_t and the original state \mathbf{z}_0 is approximately $\sqrt{\bar{\alpha}_t}$, where $\bar{\alpha}_t$ denotes the cumulative product of the noise schedule coefficients.*

proof. In the forward process, we can decompose \mathbf{z}_t as:

$$\mathbf{z}_t = \sqrt{\bar{\alpha}_t} \mathbf{z}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d), \boldsymbol{\epsilon} \perp \mathbf{z}_0. \quad (14)$$

We assume that $\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, which is reasonable since latent features in diffusion or VAE-style models are typically trained to follow a standard normal distribution. According to Proposition 1, we have $\mathbb{E}[\|z_0\|] \approx \sqrt{d}$. Therefore, the expected numerator $\mathbb{E}[\mathbf{z}_t^\top \mathbf{z}_0]$ can be computed as:

$$\begin{aligned} \mathbb{E}[\mathbf{z}_t^\top \mathbf{z}_0] &= \mathbb{E}[(\sqrt{\bar{\alpha}_t} \mathbf{z}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon})^\top \mathbf{z}_0] \\ &= \sqrt{\bar{\alpha}_t} \mathbb{E}[\|\mathbf{z}_0\|^2] + \sqrt{1 - \bar{\alpha}_t} \mathbb{E}[\boldsymbol{\epsilon}^\top \mathbf{z}_0] \\ &\approx \sqrt{\bar{\alpha}_t} d, \end{aligned} \quad (15)$$

Since \mathbf{z}_0 and $\boldsymbol{\epsilon}$ are independent and both follow $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, any linear combination of them remains Gaussian. Specifically,

$$\mathbf{z}_t = \sqrt{\bar{\alpha}_t} \mathbf{z}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \bar{\alpha}_t \mathbf{I}_d + (1 - \bar{\alpha}_t) \mathbf{I}_d) = \mathcal{N}(\mathbf{0}, \mathbf{I}_d). \quad (16)$$

According to Proposition 1, we have $\mathbb{E}[\|\mathbf{z}_t\|] \approx \sqrt{d}$. Starting from the definition of cosine similarity,

$$\begin{aligned}\mathbb{E}[\text{sim}(\mathbf{z}_t, \mathbf{z}_0)] &= \mathbb{E}\left[\frac{(\sqrt{\alpha_t} \mathbf{z}_0 + \sqrt{1-\alpha_t} \boldsymbol{\epsilon})^\top \mathbf{z}_0}{\|\mathbf{z}_t\| \|\mathbf{z}_0\|}\right] \\ &= \sqrt{\alpha_t} \mathbb{E}\left[\frac{\|\mathbf{z}_0\|^2}{\|\mathbf{z}_t\| \|\mathbf{z}_0\|}\right] \quad (\text{since } \mathbb{E}[\boldsymbol{\epsilon}^\top \mathbf{z}_0] = 0) \\ &= \sqrt{\alpha_t} \mathbb{E}\left[\frac{\|\mathbf{z}_0\|}{\|\mathbf{z}_t\|}\right].\end{aligned}\tag{17}$$

Combining these results, we obtain

$$\mathbb{E}[\text{sim}(\mathbf{z}_t, \mathbf{z}_0)] \approx \sqrt{\alpha_t},\tag{18}$$

D Additional Visual Comparisons

In this section, we provide more visual comparisons in Figs. 4–7. OSCAR consistently yields sharper textures and more faithful structures than all baselines. Traditional codecs like VVC tend to oversmooth, while HiFiC and CTC often introduce distortions in flat regions or fail to preserve edges. MS-ILLM improves local realism but suffers from blurry reconstructions. Diffusion-based methods such as CDC and PerCo generate perceptually plausible results, yet frequently introduce artifacts, color shifts, or lose semantic consistency. In contrast, OSCAR achieves a better balance between detail preservation and perceptual quality, especially under low bit-rate constraints.

E Limitations and Broader Impacts

Despite its strong performance, OSCAR still has some limitations. First, like most diffusion-based codecs, it underperforms on pixel-level distortion metrics such as MS-SSIM, primarily due to the inherent limitations of the Stable Diffusion 2.1 autoencoder, which favors perceptual quality over pixel-wise fidelity. Second, although our method supports multiple bit-rates with a single backbone, the number of bit-rates it currently handles is still limited compared to those variable-rate codecs that handle dozens of bit-rates. This is because more bit-rates mean fewer training iterations per rate under a fixed budget, and learning to denoise many different rates is inherently more difficult than focusing on just a few. Future work will aim to improve pixel-wise accuracy and further scale up the supported bit-rate range under the proposed strategy of bridging quantization and forward diffusion.

Our work introduces a simple yet effective idea that bridges quantization and diffusion, enabling single-step reconstruction across multiple bit-rates. It achieves strong perceptual quality with high efficiency. This idea opens up new possibilities for unifying compression and generation, and could inspire future research in designing more scalable diffusion-based systems for real-world applications.

References

- [1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPRW*, 2017.
- [2] Fabrice Bellard. Bpg image format. <https://bellard.org/bpg/>, 2014.
- [3] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.
- [4] Marlène Careil, Matthew J. Muckley, Jakob Verbeek, and Stéphane Lathuilière. Towards image compression with perfect realism at ultra-low bitrates. In *ICLR*, 2024.
- [5] Seungmin Jeon, Kwang Pyo Choi, Youngo Park, and Chang-Su Kim. Context-based trit-plane coding for progressive image compression. In *CVPR*, 2023.
- [6] 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*, 2025.

- [7] 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 *ICML*, 2023.
- [8] George Toderici, Wenzhe Shi, Radu Timofte, Lucas Theis, Johannes Ballé, Eirikur Agustsson, Nick Johnston, and Fabian Mentzer. Workshop and challenge on learned image compression (clic2020), 2020.
- [9] Ruihan Yang and Stephan Mandt. Lossy image compression with conditional diffusion models. In *NeurIPS*, 2023.

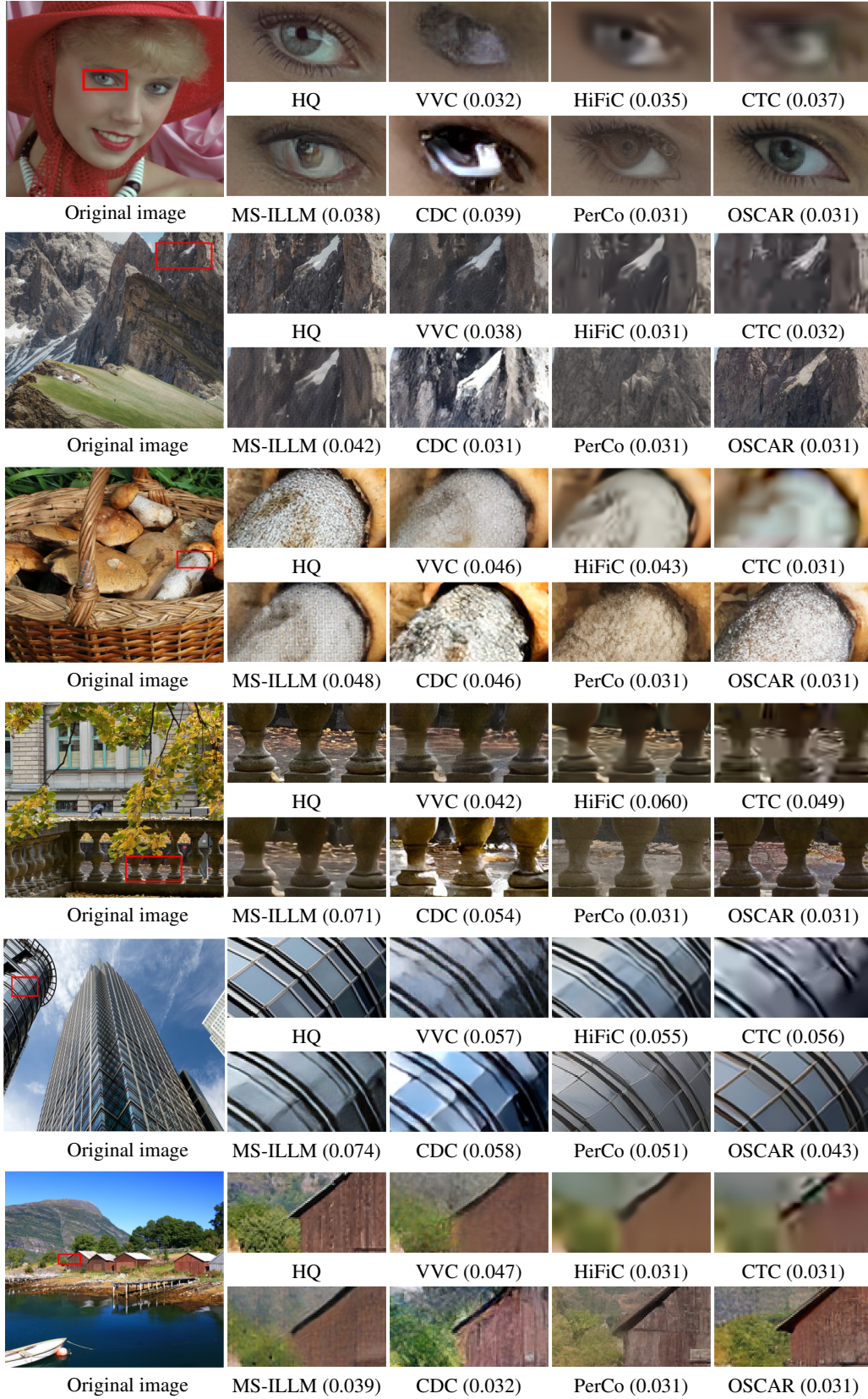


Figure 4: Additional visual comparisons. HQ denotes the original high-quality patch. The number in parentheses indicates the bits-per-pixel (bpp) used for compression by each model.

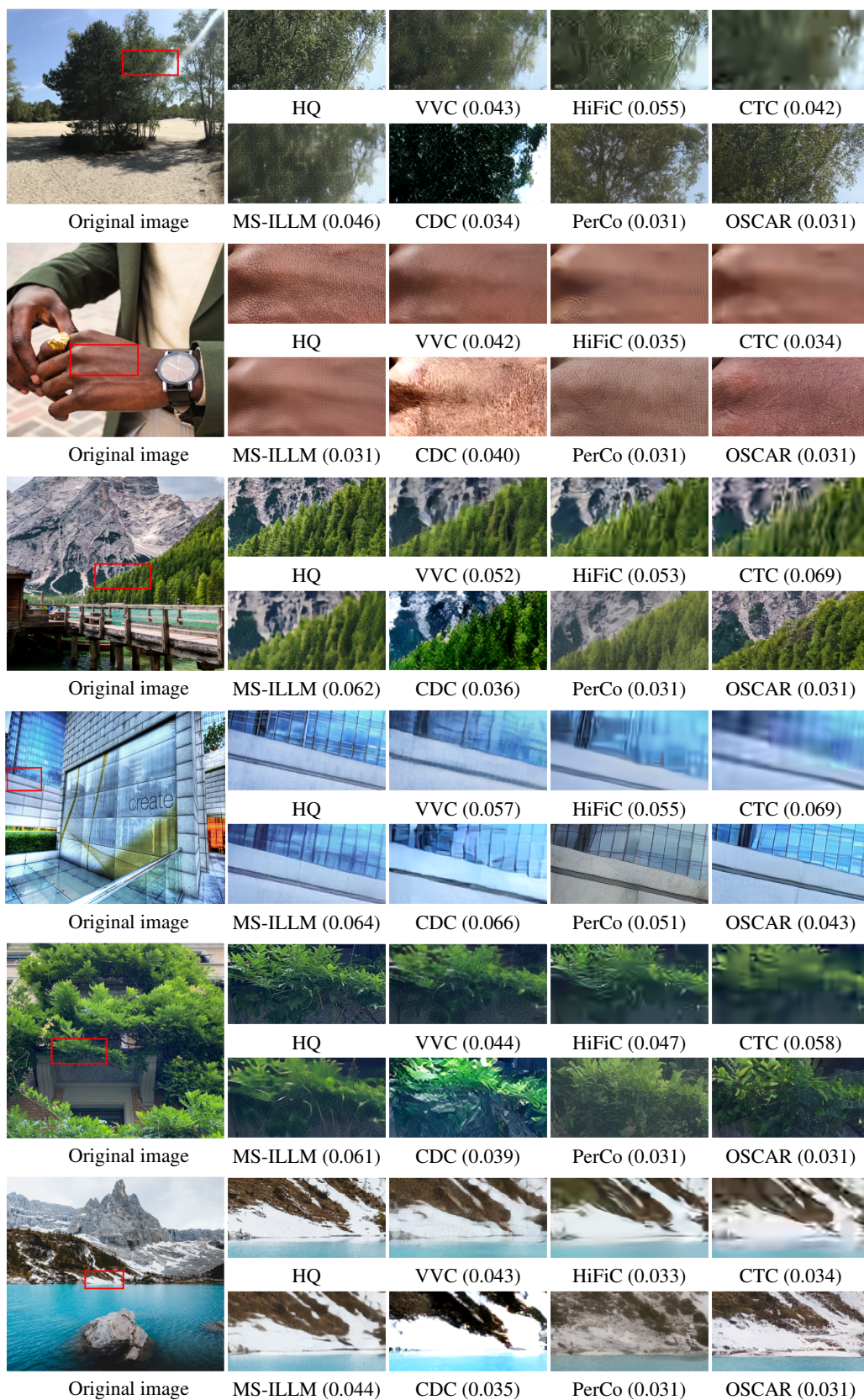


Figure 5: Additional visual comparisons. HQ denotes the original high-quality patch. The number in parentheses indicates the bits-per-pixel (bpp) used for compression by each model.

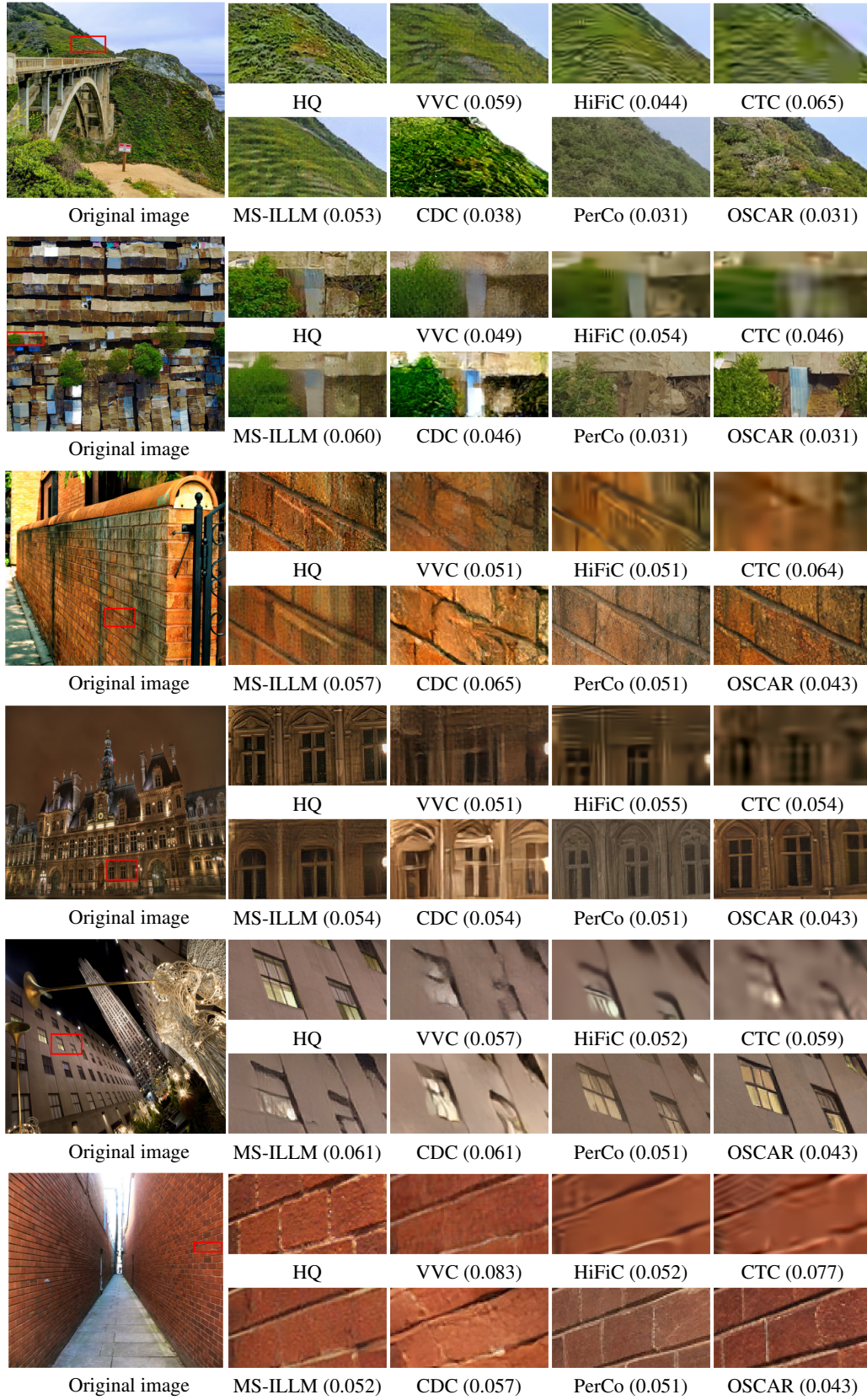


Figure 6: Additional visual comparisons. HQ denotes the original high-quality patch. The number in parentheses indicates the bits-per-pixel (bpp) used for compression by each model.



Figure 7: Additional visual comparisons. HQ denotes the original high-quality patch. The number in parentheses indicates the bits-per-pixel (bpp) used for compression by each model.

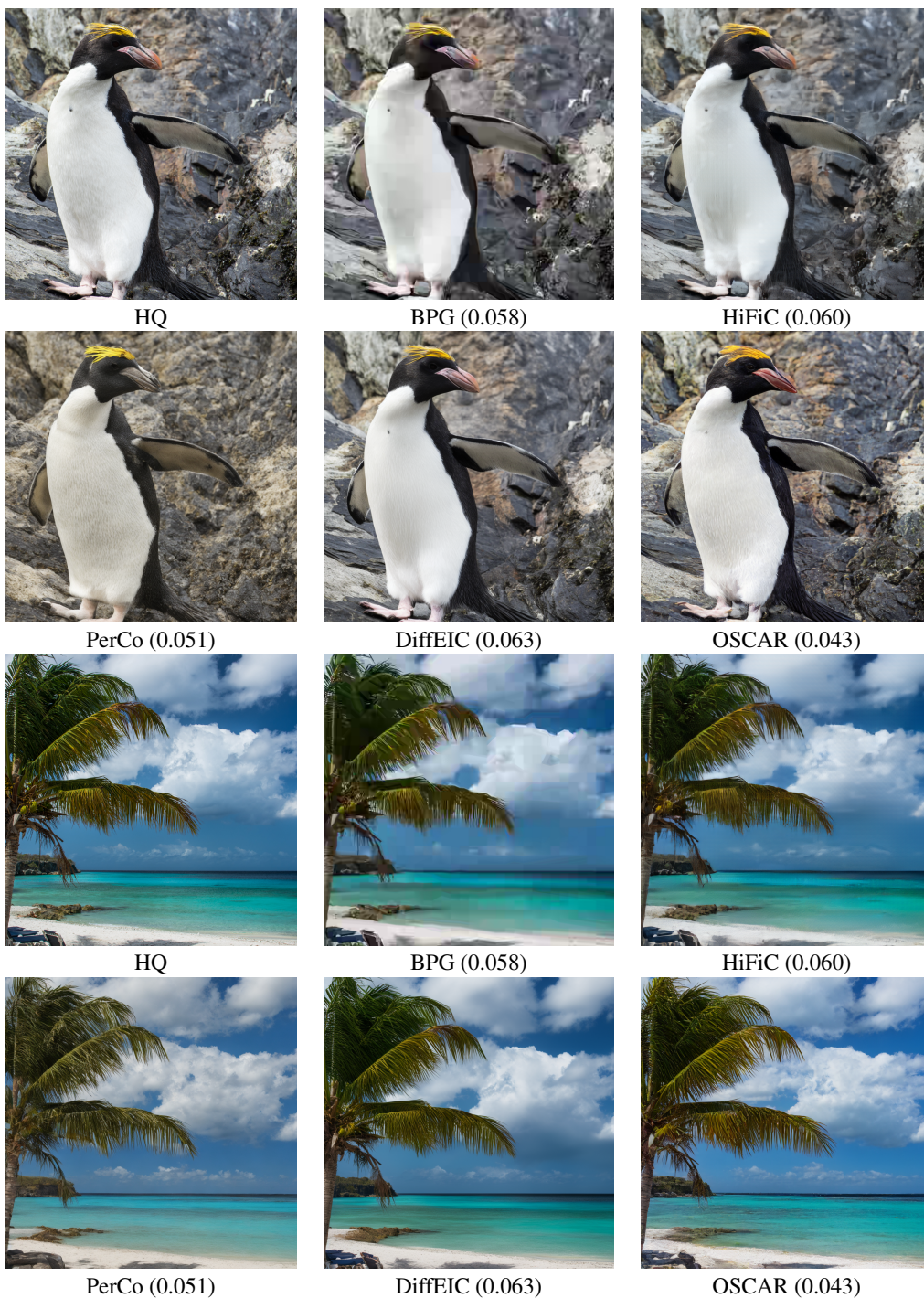


Figure 8: Additional visual comparisons. HQ denotes the original high-quality patch. The number in parentheses indicates the bits-per-pixel (bpp) used for compression by each model.

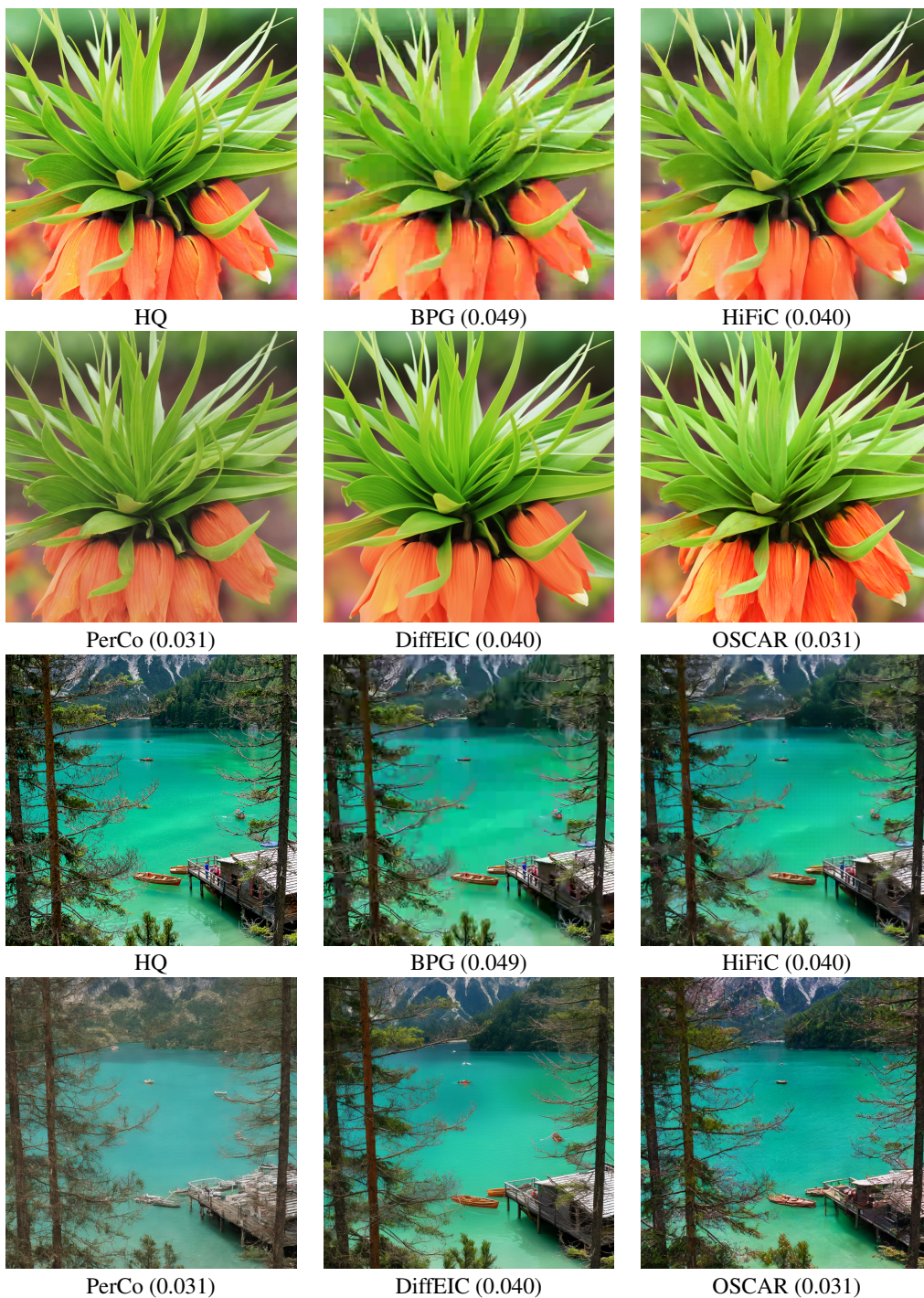


Figure 9: Additional visual comparisons. HQ denotes the original high-quality patch. The number in parentheses indicates the bits-per-pixel (bpp) used for compression by each model.