Differentiable Vector Quantization for Rate-Distortion Optimization of Generative Image Compression

Supplementary Material

A. Architecture Details

RDVQ consists of two main components: an *image tokenizer* that converts images into discrete latent tokens for representation and reconstruction, and a *masked-Transformer entropy model* that predicts their conditional distributions for rate estimation and entropy coding. We describe each component in detail below.

Image Tokenizer. The image tokenizer is a vector-quantized autoencoder adapted from the LlamaGen [59] architecture, designed to map images into discrete codebook indices while enabling high-quality reconstruction. It consists of an encoder, a decoder, and a shared codebook. To better support compression-oriented rate–distortion (RD) optimization and cross-resolution generalization, we introduce two key modifications: (i) all attention layers are removed for improved efficiency and scalability, and (ii) the original single-scale latent representation is extended to a multi-scale hierarchy.

Codebook. A shared codebook of size 4096×32 is used across all scales. The codebook is learned during a first-stage pretraining phase and then fixed during RD optimization to stabilize training.

Multi-scale representation. Instead of producing a single latent map at a fixed downsampling ratio, the encoder extracts features at three scales with downsampling factors of 16, 32, and 64. These multi-scale latents form a coarse-to-fine representation, where the lowest-resolution features capture global structure and higher-resolution features encode progressively finer details. To encourage a clean hierarchical decomposition, each higher-scale latent is defined as the residual between its feature map and the upsampled lower-scale feature.

This design not only introduces structured dependencies across scales, which benefits entropy modeling, but also increases the upper bound of achievable bitrate. For example, for a 256×256 image, the maximum bitrate under uniform coding is given by

$$\text{bpp}_{\max} = \frac{(4^2 + 8^2 + 16^2) \cdot \log_2(4096)}{256^2} \approx 0.0615, \quad (9)$$

which is higher than that of a single-scale tokenizer, enabling a wider RD operating range,.

Network architecture. Both the encoder and decoder consist of six stages of downsampling or upsampling. Each stage contains two ResBlocks followed by a downsampling or upsampling operator. The channel dimensions across stages are set to $128 \times [1, 1, 2, 2, 4, 4, 4]$. During encoding, latent features are extracted from the three lowest-resolution stages to construct the multi-scale representation, and the decoder reconstructs the image from the corresponding quantized embeddings.

Entropy Model. We employ a masked Transformer to predict a categorical distribution over codebook indices for each token, which is used for rate estimation during training and entropy coding at inference. The model operates on quantized latent tokens produced by the image tokenizer.

Architecture. The model consists of an input projection, a Transformer backbone, and an output projection. The input projection maps 32-dimensional quantized embeddings to 768-dimensional tokens. The backbone contains 12 masked Transformer layers with 8 attention heads and an MLP expansion ratio of 4. The output projection is a linear classifier over 4096 codebook entries.

Masking strategy. To model both intra-scale spatial dependencies and inter-scale hierarchical dependencies, we construct a dependency-aware autoregressive order over multi-scale latent features (coarse to fine).

As shown in Fig. 3, at each scale i, spatial positions are evenly partitioned into $n_i = 2^{(i+1)}$ segments, and each token is assigned a decoding number indicating its relative order within the scale. To enforce cross-scale dependencies, decoding numbers at finer scales are offset by the cumulative counts from all coarser scales (*e.g.*, adding n_1 for scale 2 and $n_1 + n_2$ for scale 3), ensuring that finer-scale tokens are conditioned on all coarser-scale tokens.

The decoding numbers are then flattened into a global order vector o, from which the attention mask is defined as

$$M = o > o^T, \quad (10)$$

so that each token attends only to its valid predecessors.

Tokens are reordered according to o before being fed into the Transformer. Multi-scale quantized features are rearranged to follow this order, where each token is conditioned on its preceding context: the first slice at scale i is initialized from the previous scale (or a learnable token for $i = 1$), and subsequent slices use the preceding feature as input. This ensures consistency between the input sequence and the causal mask, enabling autoregressive modeling of both intra- and inter-scale dependencies.

Resolution generalization. To support variable input resolutions, we first partition the latent feature map at each

scale into non-overlapping spatial windows (i.e., 4×4, 8×8, and 16×16 for the three scales). For each spatial location, windows from different scales that correspond to the same image region are then grouped together, and their tokens are concatenated into a single sequence for processing. Groups from different spatial locations are treated independently and batched for parallel computation, enabling flexible resolution handling while preserving aligned local multi-scale structure.

B. Training Details and Hyperparameters

Optimization objective. RDVQ is trained under a rate–distortion (RD) objective that jointly optimizes compression efficiency and reconstruction quality:

$$\mathcal{L} = \mathcal{L}_D + \lambda \mathcal{L}_R, \tag{11}$$

where $\mathcal{L}_R$ denotes the rate loss and $\mathcal{L}_D$ denotes the distortion loss.

The rate loss is defined as a cross-entropy objective between the relaxed index distribution p_{soft} and the entropy model prediction $q_{\boldsymbol{\psi}}$:

$$\mathcal{L}_R = \mathrm{CE}(p_{\text{soft}}, q_{\boldsymbol{\psi}}). \tag{12}$$

The distortion loss is composed of multiple terms to balance fidelity and perceptual quality:

$$\mathcal{L}_D = \mathcal{L}_{\text{codebook}} + \mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{LPIPS}} + 0.1\mathcal{L}_{\text{GAN}}. \tag{13}$$

Here, $\mathcal{L}_{\text{codebook}}$ denotes the vector quantization loss [14], $\mathcal{L}_{\text{MSE}}$ enforces pixel-wise reconstruction accuracy, $\mathcal{L}_{\text{LPIPS}}$ [70] improves perceptual similarity, and $\mathcal{L}_{\text{GAN}}$ [18] further enhances visual realism.

Training pipeline. To obtain a high-quality codebook and stabilize rate–distortion (RD) optimization, we adopt a three-stage training pipeline, where different objectives are applied progressively.

In the first stage, we pretrain the image tokenizer using only the distortion objective $\mathcal{L}_D$. This stage focuses on learning a representative codebook and high-quality reconstruction without considering compression efficiency.

In the second stage, we train the entropy model separately using only the rate objective $\mathcal{L}_R$. With the tokenizer and codebook fixed, this stage learns to accurately model the distribution over codebook indices, providing reliable probability estimates for subsequent RD optimization.

In the final stage, we perform joint RD optimization using the full objective $\mathcal{L} = \mathcal{L}_D + \lambda\mathcal{L}_R$, while keeping the codebook fixed. Models at different bitrate levels are obtained through a staged λ-based curriculum, which enables smooth adaptation across rate regimes.

After training on ImageNet, all models are further fine-tuned on higher-resolution datasets to improve perceptual quality and generalization.

Implementation details. We train RDVQ on ImageNet [57], OpenImage [30], and DF2K [1] datasets. ImageNet is used for large-scale low-resolution pretraining, while OpenImage and DF2K are employed for high-resolution fine-tuning to improve cross-resolution generalization during inference.

Training stages. The image tokenizer is first pretrained on 256×256 ImageNet patches for 7×10^5 iterations (batch size 32, learning rate 1×10^{-4}, BF16). The learned codebook is then frozen. Next, the entropy model is pretrained on 256×256 ImageNet patches for 4×10^5 iterations (batch size 80) to provide stable probability estimation. Finally, joint RD optimization is performed with the full objective $\mathcal{L} = \mathcal{L}_D + \lambda\mathcal{L}_R$ and the codebook fixed. Models are first trained at relatively high bitrates and then progressively fine-tuned toward lower bitrates. From this stage onward, all models are trained in FP32 precision for stable optimization.

High-resolution fine-tuning. Models trained solely on ImageNet 256×256 patches generalize poorly to high-resolution datasets (DIV2K, CLIC2020). To improve cross-resolution performance, we fine-tune models on OpenImage (512×512, batch size 4, lr 1×10^{-5}, 4×10^5 iterations) followed by DF2K (random crops from 512 to 2048, batch size 1, lr 5×10^{-6}). This strategy enables the model to adapt to varying resolutions, improving reconstruction quality on high-resolution images. Comparative results on DIV2K show progressive improvements from ImageNet-only, OpenImage, to DF2K fine-tuning (see Fig. S1).

All primary experiments use four NVIDIA RTX 4090 GPUs; DF2K high-resolution fine-tuning is performed on a single NVIDIA RTX Pro 6000 GPU.

RD trade-off and temperature. The rate loss is defined as the cross-entropy between the relaxed index distribution p_{soft} and the entropy model prediction $q_{\boldsymbol{\psi}}$, with the softmax temperature τ controlling distribution sharpness. Smaller τ produces sharper p_{soft}, resulting in concentrated gradients and stronger effective rate constraint for a given λ, whereas larger τ smooths p_{soft}, producing more diffuse gradients and weaker rate constraints. Thus, τ and λ must be jointly considered to achieve the desired rate–distortion trade-off.

Following empirical observations, we choose τ and λ based on the bitrate regime. In the ultra-low bitrate regime (bpp < 0.025), smoother relaxation ($\tau = 0.1$) tends to yield better RD performance, whereas in higher bpp regimes (bpp > 0.025), sharper relaxation ($\tau = 0.01$) generally performs better. Correspondingly, λ is set to $\{4.8, 7.2, 12\}$ for low-bitrate models ($\tau = 0.1$) and $\{0.8, 1.2\}$ for higher-bitrate models ($\tau = 0.01$) to maintain reasonable rate constraints.

The necessity of this piecewise strategy is confirmed by comparing it to a fixed-sharp relaxation baseline ($\tau = 0.01$, $\lambda = \{0.8, 1.2, 1.8, 2.4\}$). As shown in Fig. S2, the piecewise strategy improves RD performance at ultra-low bpp,

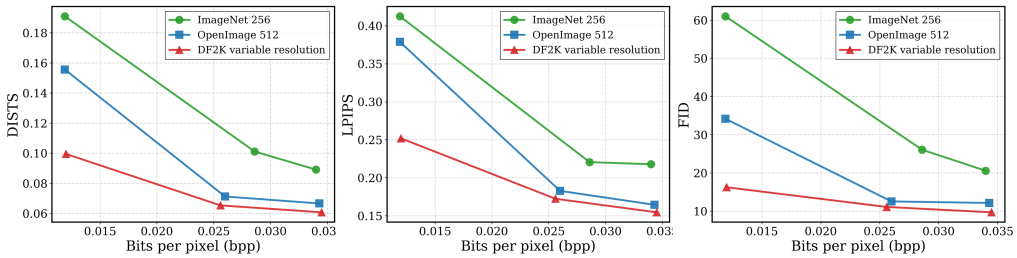

Figure S1. Effectiveness of tuning on high-quality images (evaluated on Div2K validation set).

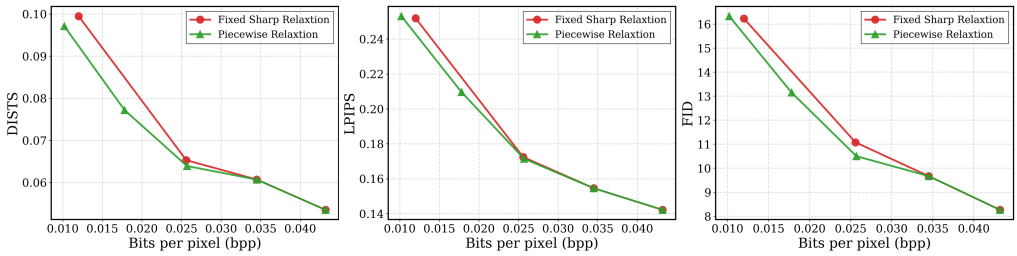

Figure S2. Effectiveness of piecewise strategy (evaluated on Div2K validation set).

highlighting the effectiveness of piecewise relaxation.

C. Experiments

C.1. Evaluation of third-party models.

We compare RDVQ against recent state-of-the-art open-source learned image compression methods, including DDCM [52], RDEIC [38], ResULIC [28], MS-ILLM [51], PerCo [8], OSCAR [19], DLF [66], and StableCodec [71], as well as the classical codec VVC [7]. For DDCM, RDEIC, MSILLM, PerCo, OSCAR, DLF, StableCodec, and VVC, we run the official implementations under the same evaluation protocol as our RDVQ. For ResULIC, we report the performance provided in the official release, since their evaluation is conducted on the same benchmark settings.

C.2. Evaluation Details

For FID evaluation on CLIC2020 and DIV2K, following HiFiC [48], we compute FID on patches rather than full images. Specifically, each input image is divided into overlapping 256×256 patches with a stride of 128, and FID is computed over the distribution of these patches.

C.3. Quantitative Results

Figure S3, Fig. S4, and Fig. S5 present RD curves on CLIC2020-test, DIV2K-Val, and Kodak, respectively, using a pixel-level distortion metric. These plots provide a more detailed evaluation of RDVQ across different datasets.

C.4. Test Time Rate Control Examples

RDVQ enables test-time bitrate adjustment within a limited range via prefix transmission and autoregressive completion, without retraining. As shown in Fig. S6 and Fig. S7, the model can smoothly vary the rate within 0.05–0.3 bpp on CLIC2020-test and DIV2K-Val, while maintaining high perceptual quality with no noticeable visual degradation.

C.5. Additional Visual Examples

We provide additional qualitative comparisons on CLIC2020-test in Figs. S8–S11 and Figs. S12–S14. The former corresponds to relatively higher bitrates (bpp $= 0.025 \sim 0.045$), while the latter shows more aggressive compression (bpp $= 0 \sim 0.02$). In both cases, RDVQ consistently yields sharper details and better structural fidelity than competing methods, supporting the quantitative results.

C.6. Comparison with other VQ-based methods

We further compare our method with several VQ-based approaches [33, 54, 67] on Kodak. As shown in Tab. S1, our method consistently outperforms all baselines in BD-DISTS, demonstrating superior rate–distortion efficiency.

Method	Ours	CGIC [33]	Diffo [54]	OneDC [67]
BD-DISTS (%, Kodak)	**0.0**	+5422.8	+207.11	+18.81

Table S1. Comparison with additional VQ-based methods on the Kodak dataset in terms of BD-DISTS. Lower is better.

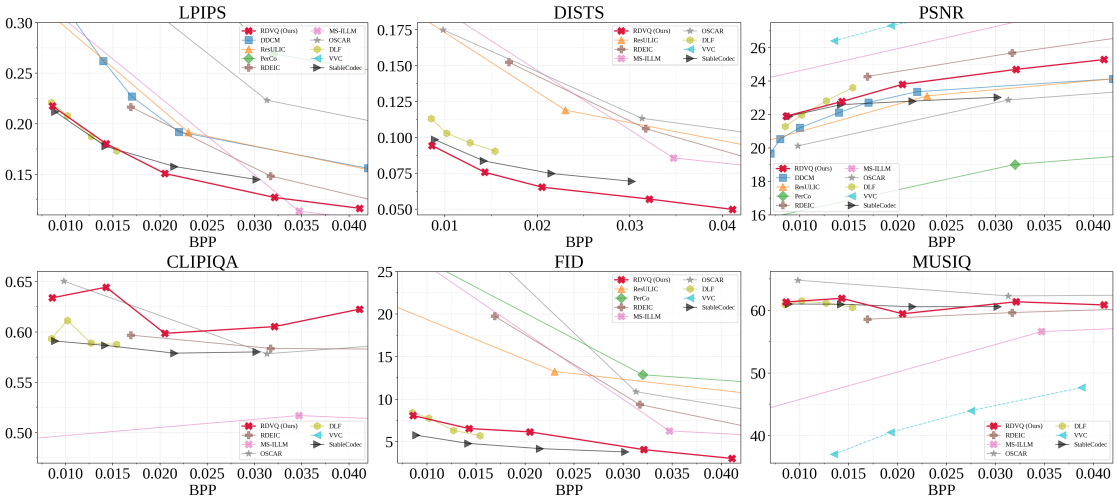

Figure S3. Rate-distortion curves on the CLIC 2020 testset.

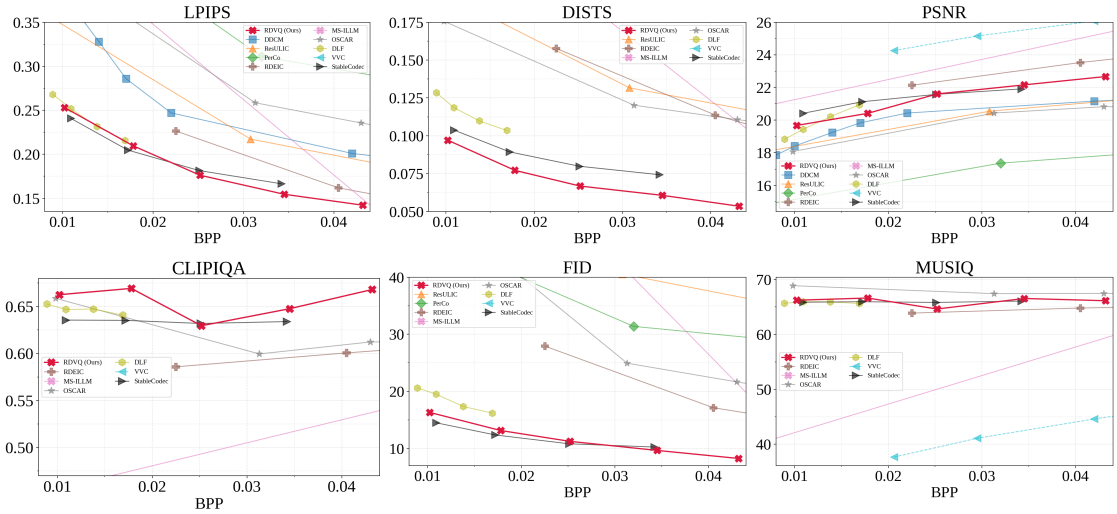

Figure S4. Rate-distortion curves on the DIV2K-val dataset.

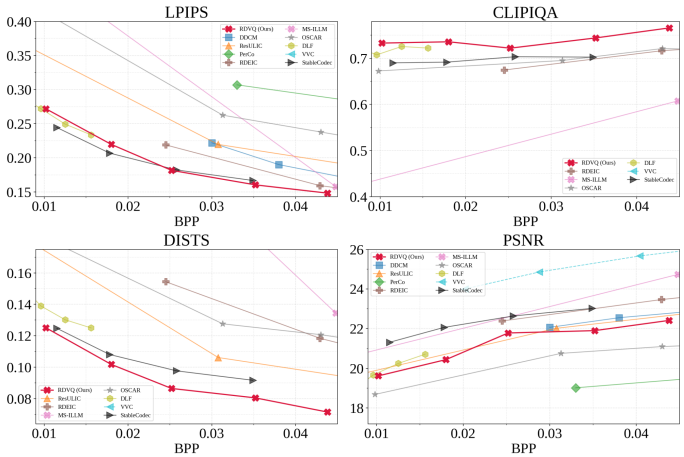

Figure S5. Rate-distortion curves on the Kodak dataset.

Figure S6. Test time rate control examples on CLIC2020 testset.

Figure S7. Test time rate control examples on DIV2K-val dataset.

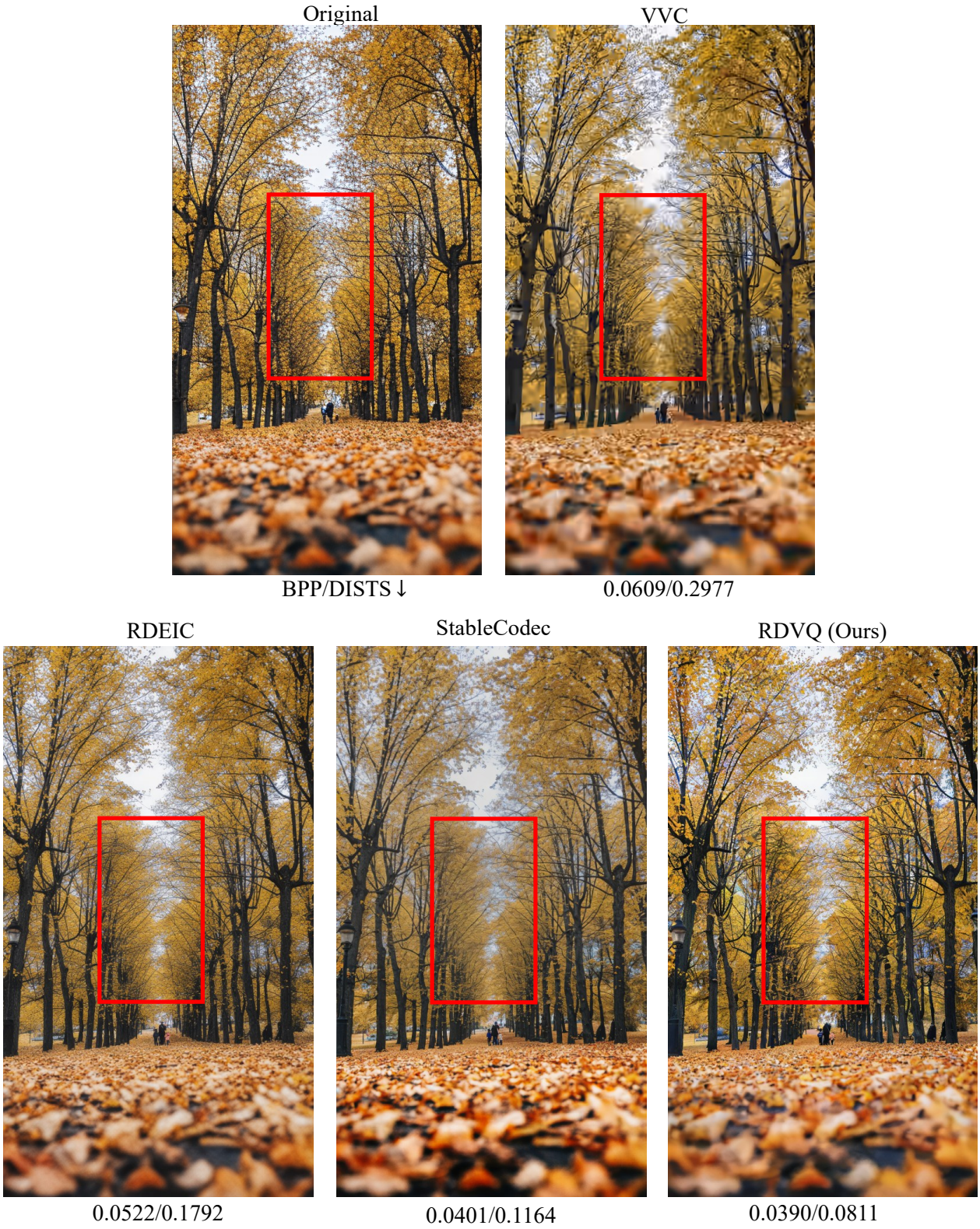

Figure S8. Visual examples and comparisons on 2K-resolution images from CLIC2020-test dataset.

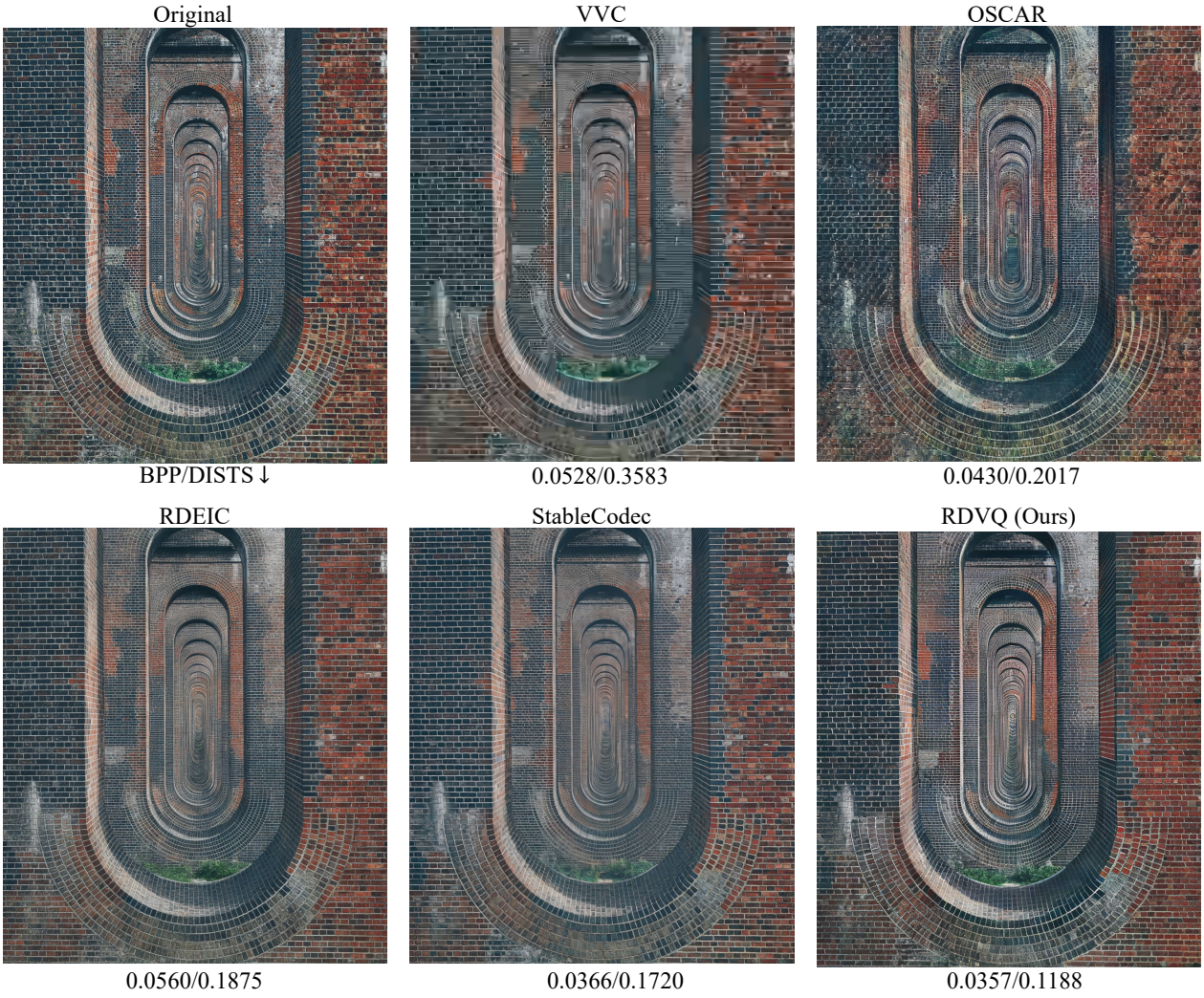

Figure S9. Visual examples and comparisons on 2K-resolution images from CLIC2020-test dataset.

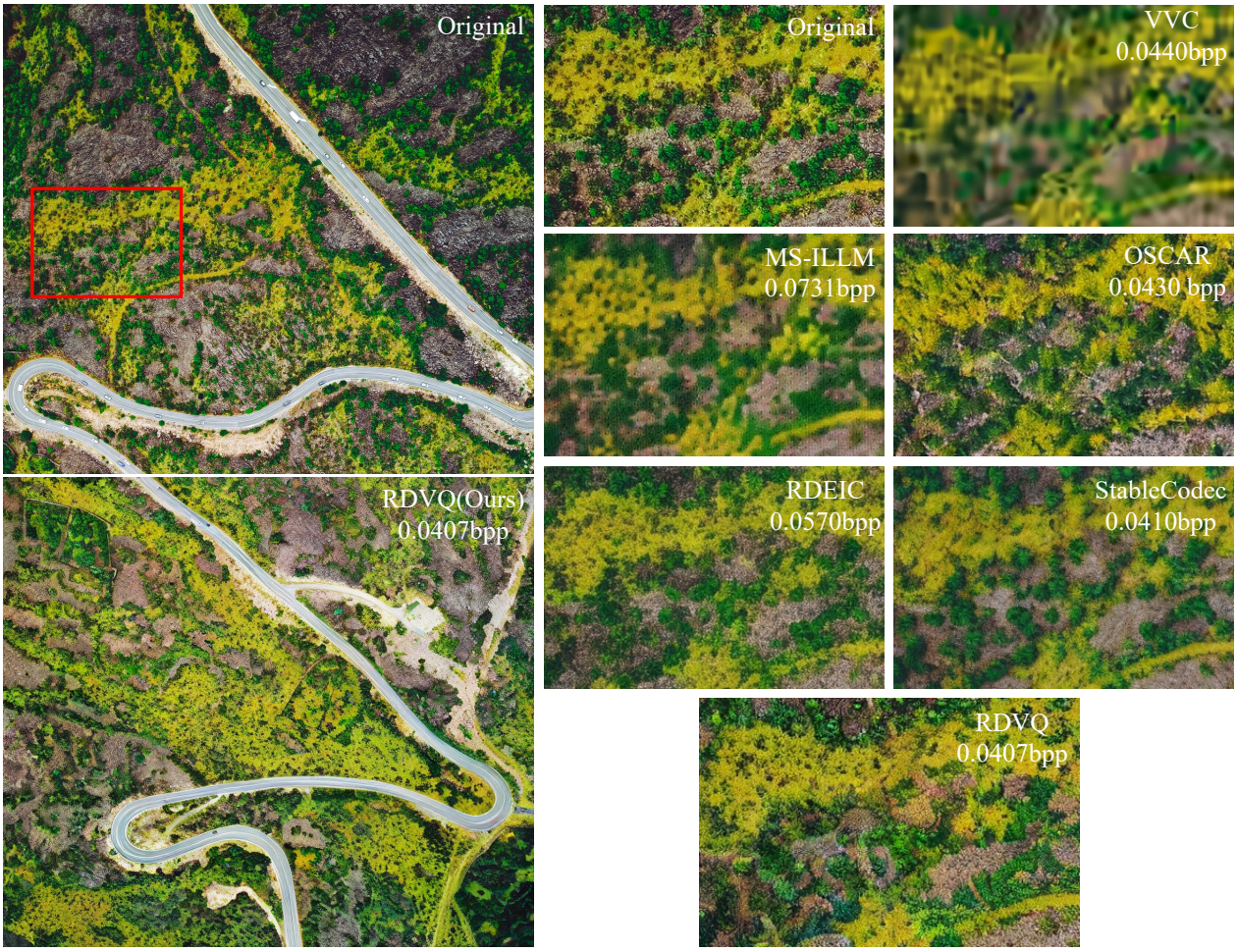

Figure S10. Visual examples and comparisons on 2K-resolution images from CLIC2020-test dataset.

Figure S11. Visual examples and comparisons on 2K-resolution images from CLIC2020-test dataset.

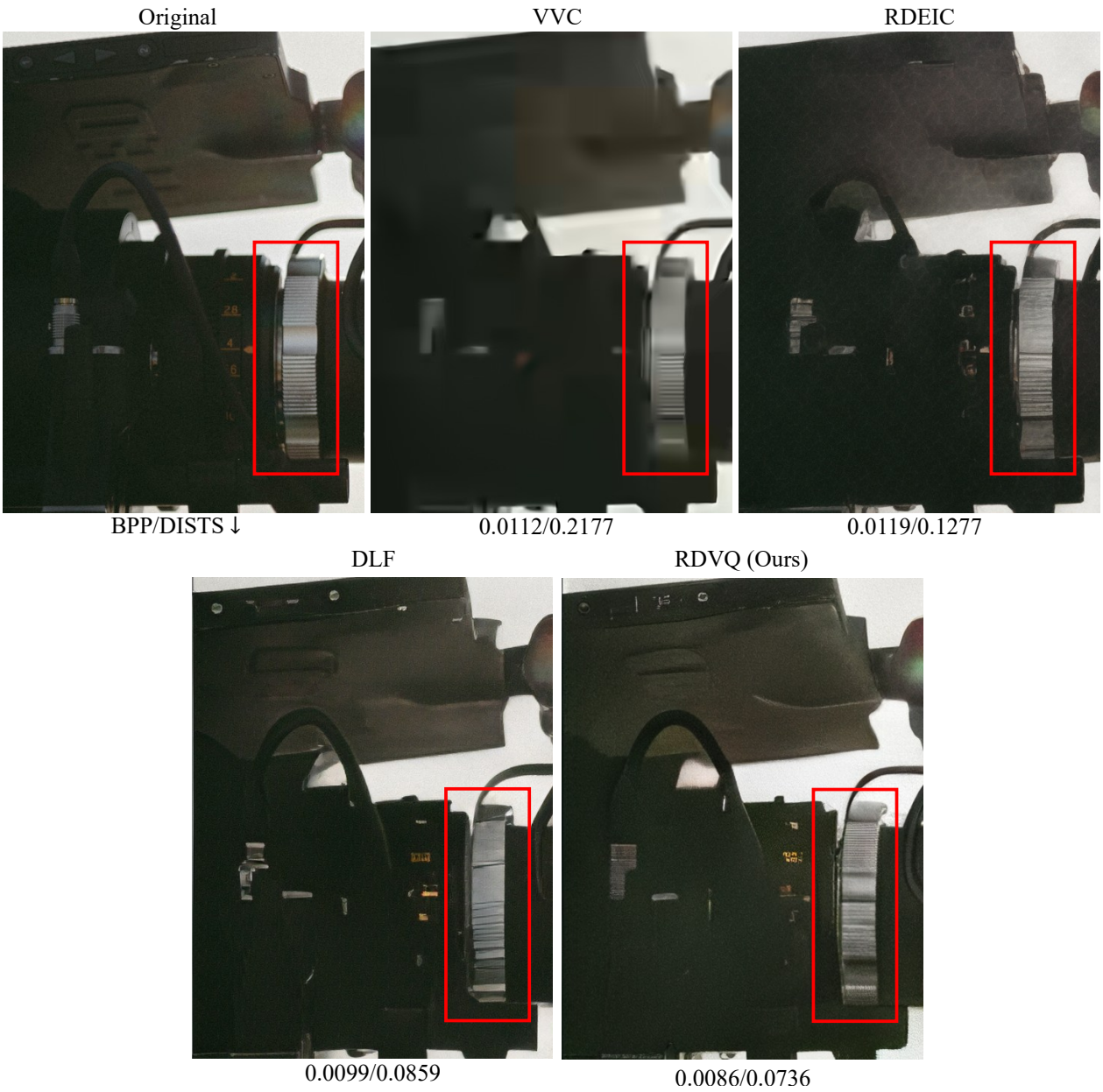

Figure S12. Visual examples and comparisons on 2K-resolution images from CLIC2020-test dataset.

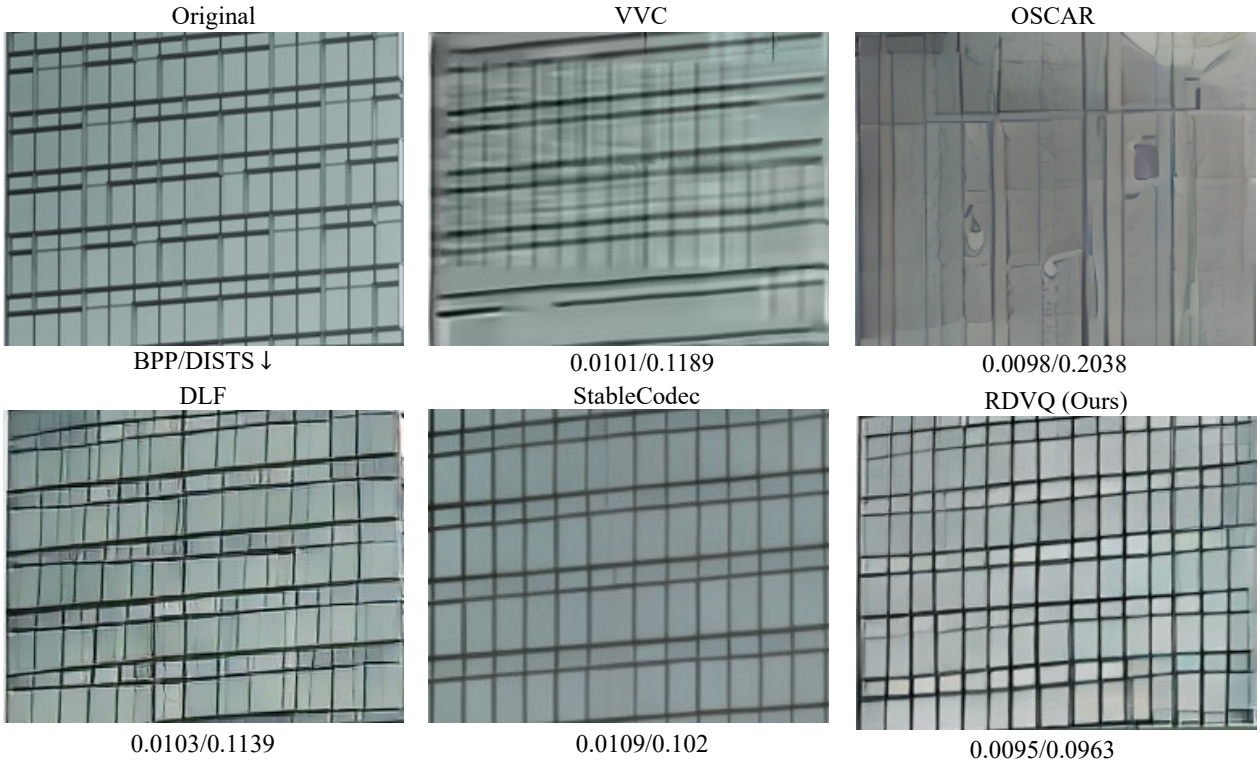

Figure S13. Visual examples and comparisons on 2K-resolution images from CLIC2020-test dataset.

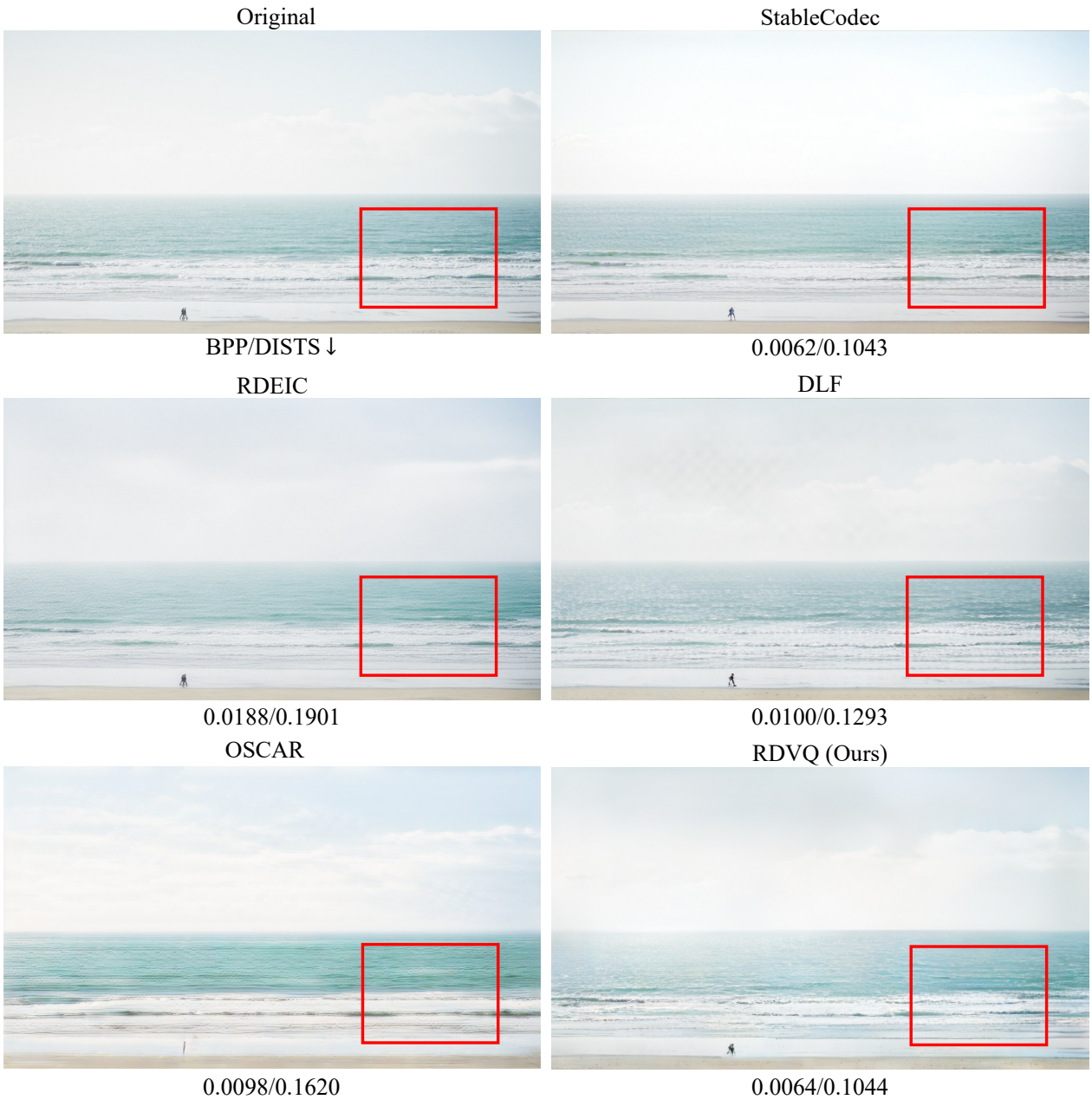

Figure S14. Visual examples and comparisons on 2K-resolution images from CLIC2020-test dataset.