Learned Image Compression with Discretized Gaussian Mixture Likelihoods and Attention Modules: Supplementary Material

Zhengxue Cheng[1], Heming Sun[2,3], Masaru Takeuchi[2], Jiro Katto[1]
[1] Department of Computer Science and Communication Engineering, Waseda University, Tokyo, Japan
[2] Waseda Research Institute for Science and Engineering, Tokyo, Japan
[3] JST, PRESTO, 4-1-8 Honcho, Kawaguchi, Saitama, Japan
{zxcheng@asagi., hemingsun@aoni., masaru-t@aoni., katto@}waseda.jp

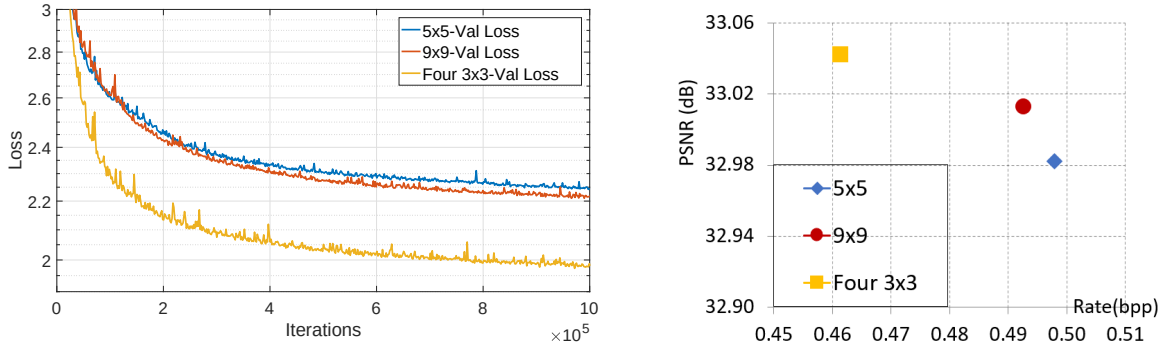
The supplementary materials give some ablation study and more results, which are not included in the submitted paper due to page limitation.

1. Ablation Study on Network Architecture

1.1. Backbone

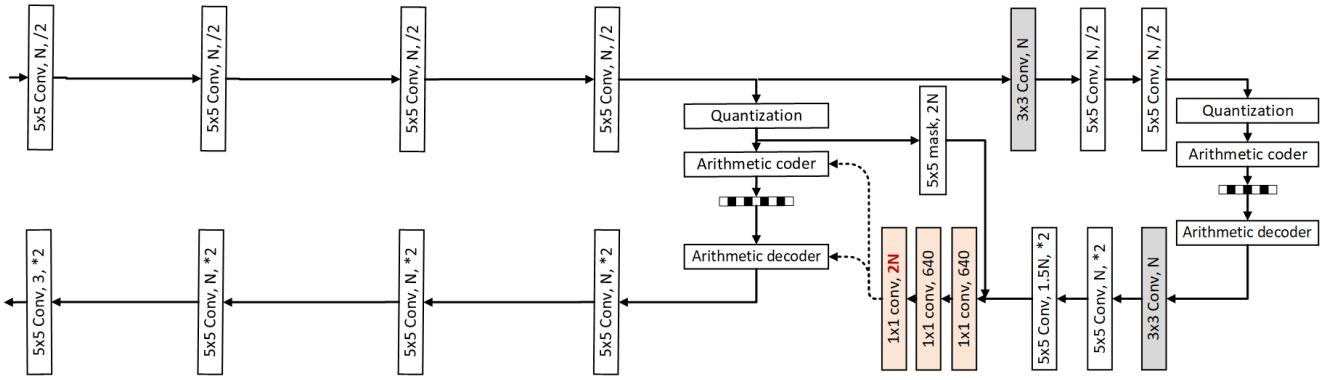
In this paper, we use a similar structure as [1] in Fig. 2, whose original idea is from [2]. Four stacked 3×3 convolutions with residual connection can achieve larger receptive field with fewer parameters than one convolution with 5×5 kernel, which was used in the works [3, 4, 5]. On the other hand, subpixel convolution is used to replace commonly used transposed convolution as upsampling units to keep more details at the decoder side.

To illustrate the ablation study on network architecture, we have conducted experiments by comparing three cases: (a) 5×5 kernel for each downsampling operation as [3, 4, 5] as Fig. 2(a); (b) 9×9 kernel for each downsampling operation in the main autoencoder as Fig. 2(b). The kernel size in auxiliary autoencoder has fewer effect, so they are kept as 5×5; (c) The network we used (denoted as our anchor) as Fig. 2(c): four 3×3 kernels with residual connection for each downsampling operation and use subpixel convolution in synthesis transform. Except the network architecture, the other training settings are kept the same for the above three cases. The case (b) was tested, because it has the same architecture with (a), but has the same receptive field as (c). Gaussian

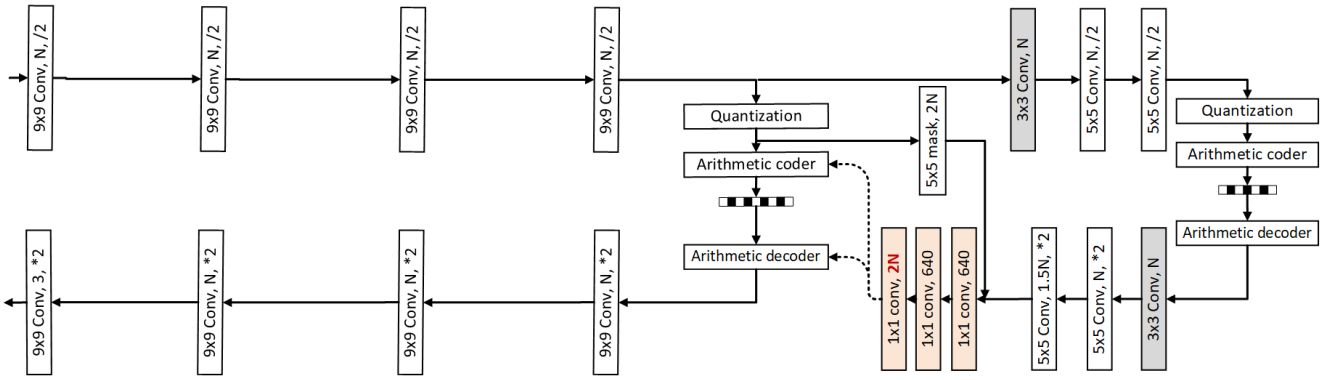


(a) Asymptotic performance (b) Rate-distortion performance at 1×10^6 iterations

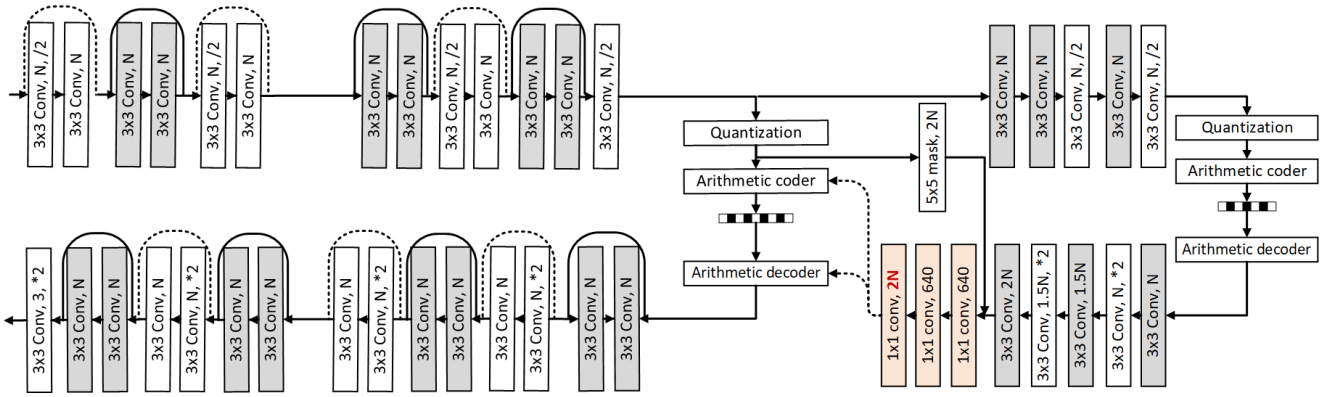
Figure 1: The ablation study on network architecture with $N = 128$, optimized by MSE with $\lambda = 0.015$.



(a) 5×5 kernel for each downsampling operation as [3, 4, 5]



(b) 9×9 kernel for each downsampling operation in the main autoencoder



(c) Four 3×3 kernels for each downsampling operation in the main autoencoder

Figure 2: The ablation study on network architectures with single Gaussian entropy model (K=1).

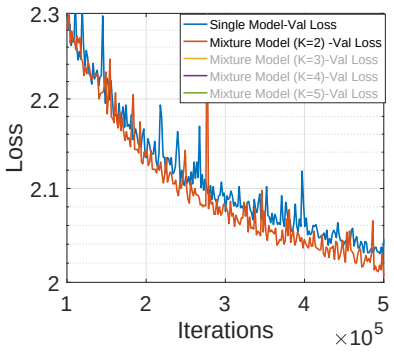
mixture model is not incorporated, so single Gaussian model is used, requiring $2 \times N$ channels for the output of entropy models. The number of filters N is equal to 128.

The loss curve and rate-distortion performance are shown in Fig. 1, respectively. It can be observed our network achieves better coding efficiency than the network architecture of [3, 4, 5], by reducing the rate about 6% ($6\%=\frac{0.49\text{bpp}-0.46\text{bpp}}{0.49\text{bpp}}$) with even slightly better quality. Therefore, we used the network architecture of Fig. 2(c) as the *Anchor* in the Section 5.1 of our paper. One thing to note, the RD points in Figure 6(b) of original paper are

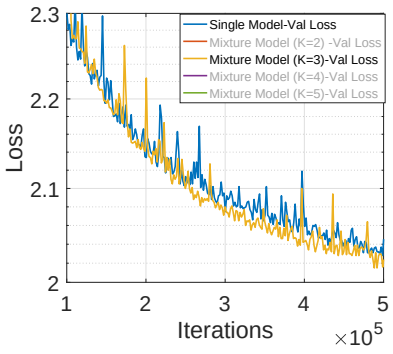
slightly different from Fig. 1(b) only because all the RD points in Figure 6(b) are tested at about 4×10^5 iterations, and all the RD points in Fig. 1(b) are tested at about 1×10^6 iterations for fair comparison. The coding gain of our strategies always exist, and the asymptotic loss curves can illustrate this difference clearly.

1.2. The Number of Mixtures K

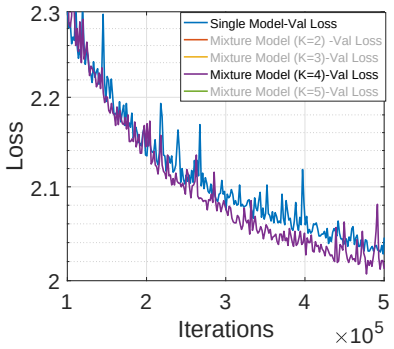
In the paper, we used $K = 3$ empirically. To validate the effect of the number of mixtures K, we tested the performance using K in the set of $\{2, 3, 4, 5\}$ and results are discussed in Fig. 3. The loss values of mixture models are smaller than the loss of single Gaussian model, but the performance gain of different cases seems quite similar. Besides, we draw the RD points as shown in Fig. 3(e). It can be observed that when K is equal to $\{3, 4, 5\}$, the performance almost saturates. We can not observe more coding gain by increasing the number of mixtures.



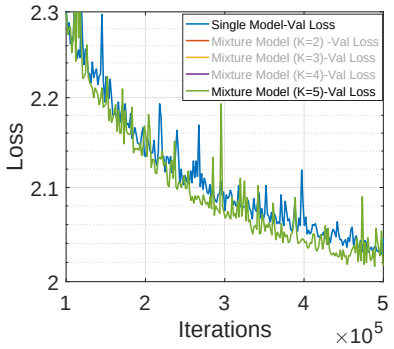
(a) Loss curves of single model and $K = 2$



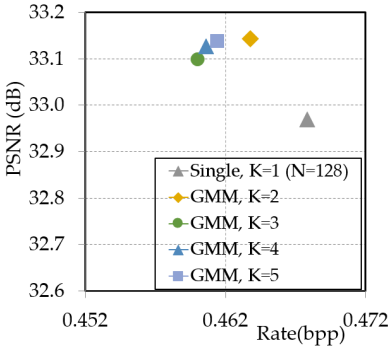
(b) Loss curves of single model and $K = 3$



(c) Loss curves of single model and $K = 4$



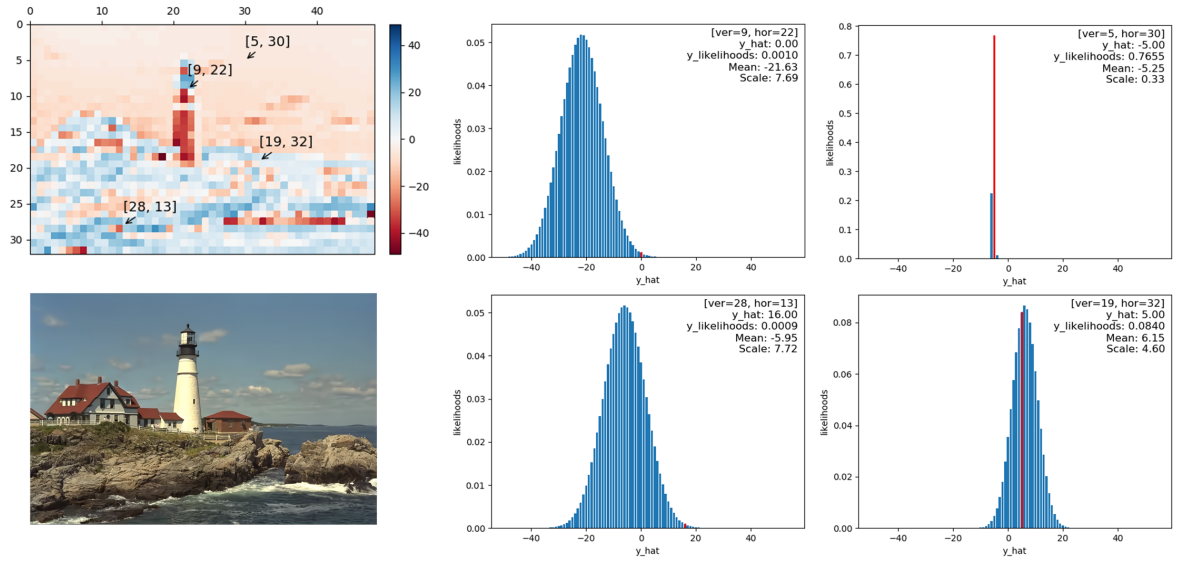
(d) Loss curves of single model and $K = 5$



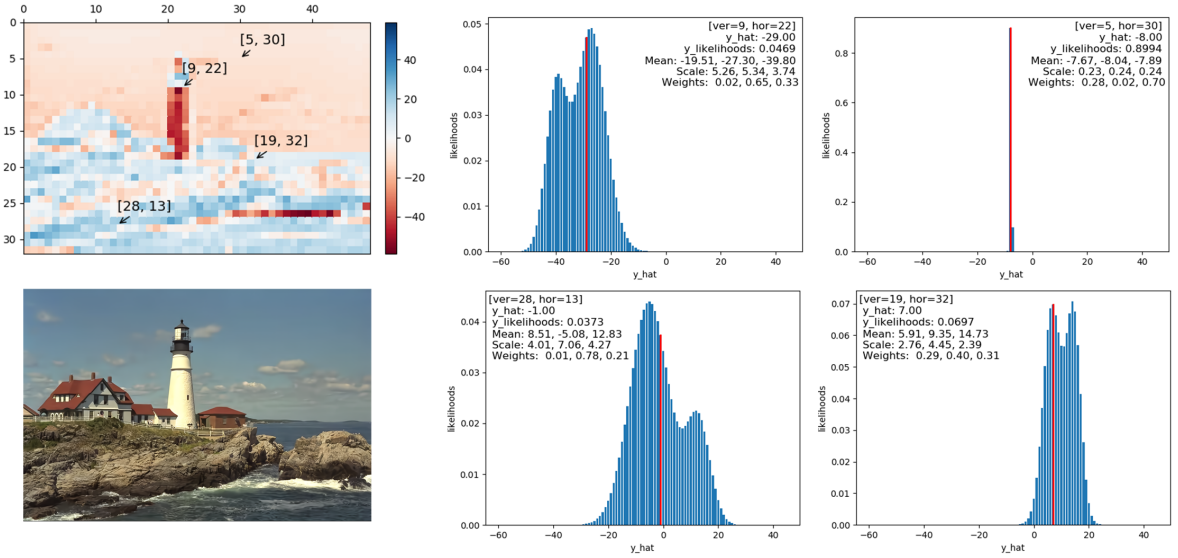
(e) Rate-distortion performance

Figure 3: The ablation study on the number of mixtures K with $N = 128$, optimized by MSE with $\lambda = 0.015$.

Moreover, to show the difference of single Gaussian distribution and mixture Gaussian distributions, we visualize the estimated likelihoods for some locations in the image *Kodim21* from Kodak dataset in Fig. 4. The size of *Kodim21* is 512×768, so the size of compressed codes $\hat{\boldsymbol{y}}$ is 32×48 after four downsampling operations in analysis transform. We select four representative locations (denoted as [Vertical axis, Horizontal axis]), that is, $[5, 30]$ in the sky, $[9, 22]$ at the white tower, $[28, 13]$ around the rock, $[19, 32]$ between the boundaries. Single Gaussian model can only achieve symmetric and fixed shape in terms of discrete distribution, as shown in Fig. 4(a), while our proposed Gaussian mixture models achieves more flexible and arbitrary shapes in terms of discrete distribution, as shown in Fig. 4(b). It is noted that in many cases of simple regions, our model can degrade to single model distribution when three estimated mean values are the same, such as the location $[5, 30]$.



(a) Single Gaussian Model



(b) Gaussian Mixture Model

Figure 4: Visualization of estimated distributions for latent codes.

2. Test Methodology on Codecs

2.1. Versatile Video Coding (VVC)

In order to test the performance of VVC, we used the VVC official test model VTM 5.2 [1]. However, the dataset we used are RGB format, instead of YUV format, which are widely used in traditional compression standards. According to the document of VVC, given a RGB image, we convert RGB888 to YUV444 or YUV420 format using the definition from [6]. Then, the command line for compressing the given YUV files *ImageYUV444.yuv* is

[1] https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/tree/VTM-5.2, accessed on July 17, 2019

EncoderApp -c encoder_intra_vtm.cfg -i ImageYUV444.yuv -b ImageBinary.bin -o ImageRecon.yuv -f 1 -fr 2 -wdt ImageWidth -hgt ImageHeight -q QP --OutputBitDepth=8 --OutputBitDepth=8 --OutputBitDepthC=8 --InputChromaFormat=444

where *encoder_intra_vtm.cfg* is the official intra configuration files by default. *-f* denotes the number of frames to encode. VVC requires the frame rate must be larger than 1, so we set *-fr* as 2, and we only have 1 frame, so it did not affect the performance. *-wdt* and *-hgt* specify the image size. *-q* specify the quantization parameters, and we use the QP in the set of $\{22, 27, 32, 37, 42, 47\}$. And all the YUV components have 8 bits. The color format is 444. On the other hand, YUV420 is more common than YUV444 in compression standards for a long history, so we also compress the image *ImageYUV420.yuv* using the command line as

EncoderApp -c encoder_intra_vtm.cfg -i ImageYUV420.yuv -b ImageBinary.bin -o ImageRecon.yuv -f 1 -fr 2 -wdt ImageWidth -hgt ImageHeight -q QP --OutputBitDepth=8 --OutputBitDepth=8 --OutputBitDepthC=8 --InputChromaFormat=420

The results are shown in Fig. 5. We can observe that the performance of YUV420 are worse than that of YUV444 due to some sampling loss of chroma components, especially decreasing the quality at high rate. Therefore, we used YUV444 for comparison in our paper.

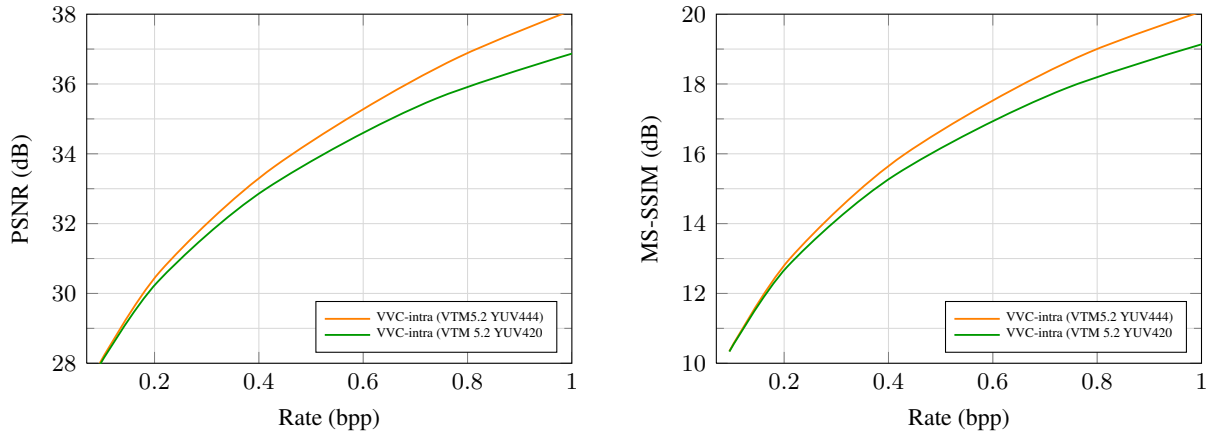


Figure 5: Performance Comparison of VVC codecs on Kodak dataset.

2.2. Boundary Handling

For both traditional coding standards and deep learning based compression algorithms, boundary handling needs to be considered when the input images have arbitrary sizes. The CLIC validation images have arbitrary heights and widths. Therefore, for VVC the input size must be a multiple of the minimum CU size according to the specification of VVC encoder. Our solution is to pad the height and width of images to a multiple of 8 using reflect padding. We also encode the height and width of input images into the bitstream and crop the required part to get the original size after decoding.

For our learned codec, the height and width of input images must be at least a multiple of 64, because the minimum size in the networks are $[\frac{H}{64}, \frac{W}{64}]$ when the input size is $[H, W]$. Similarly, we pad the image height and width to a multiple of 64 using reflect padding before feeding the data into the our learned neural network models, and encode the height and width into the bitstream. After the decoding, we crop the valid part to reconstruct the images with original size.

3. More Qualitative Results

To demonstrate our method can generate more visually pleasant results, more qualitative comparisons using the images from Kodak [7] and CLIC validation dataset [8] are given in the following.

Fig. 6 shows reconstructed images *kodim01* with approximately 0.15 bpp and a compression ratio of 160:1. The latch on the door is only maintained well for our proposed model optimized by MS-SSIM with similar bit rate. However, the reconstructed images for our method optimized by MSE and VVC look similarly blurry. For the other codecs, such as HEVC (BPG), JPEG2000 and JPEG, clear blocking artifacts appear.

Fig. 7 shows reconstructed images *kodim17* with approximately 0.10 bpp and a compression ratio of 240:1. The bit rate of proposed method optimized by MSE is a little bit higher than others, because we only train discrete λ for six models, which can not target at 0.10 bpp accurately. The reconstructed images by our proposed method optimized by MS-SSIM appears more natural and preserves more details, including the face of statue and the textures of the ball. Therefore, our method achieves slightly better subjective quality results than other codecs.

Fig. 8 shows reconstructed images *kodim19* with approximately 0.12 bpp and a compression ratio of 200:1. It can be observed that the smile face appears more natural and the angle of mouth are better preserved in our reconstructed images using MS-SSIM-optimized model. Besides, the sky and grass reconstructed by our proposed method are also much better than other reconstructed images.

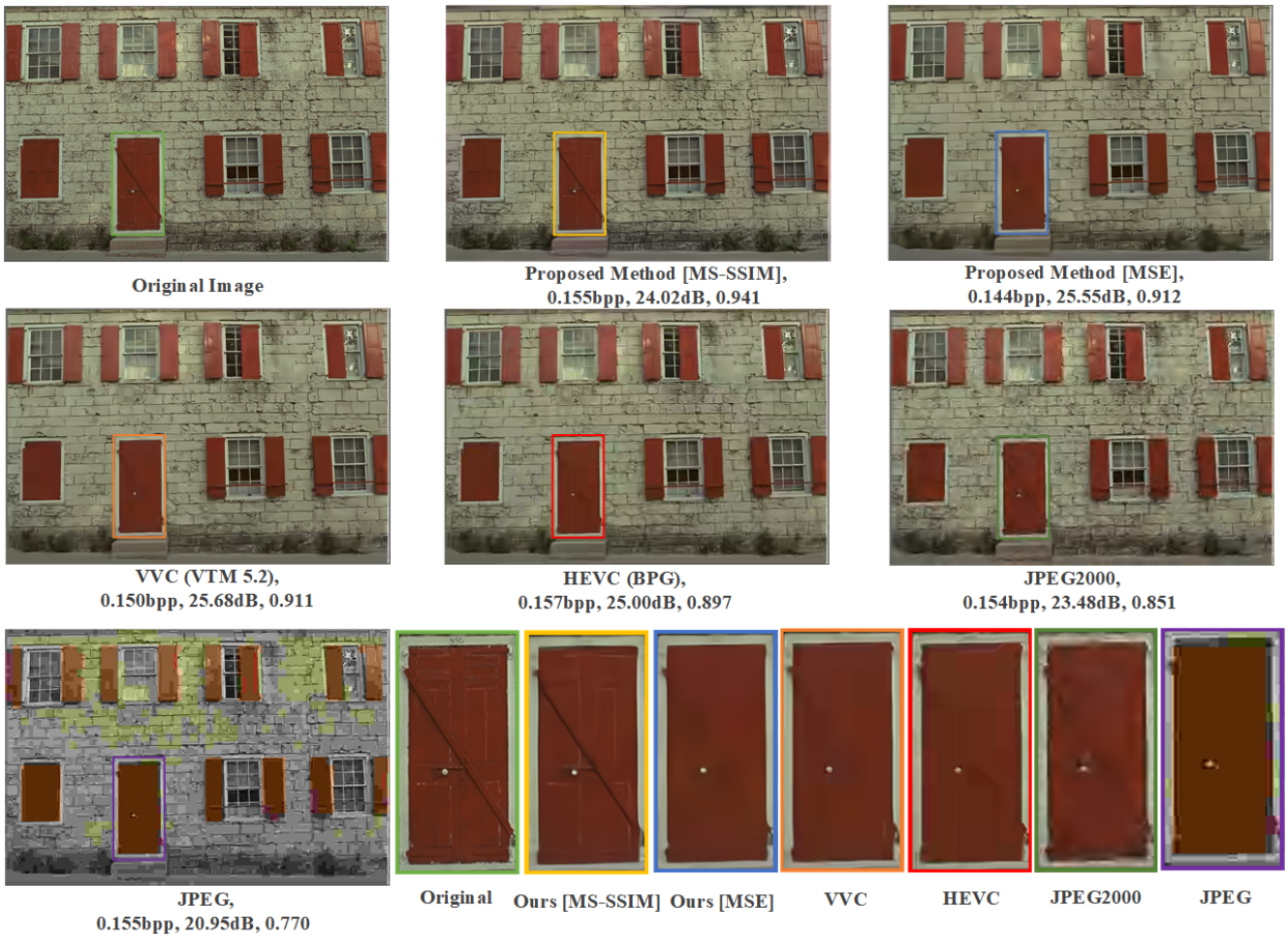


Figure 6: Visualization of reconstructed images *kodim01* from Kodak dataset.

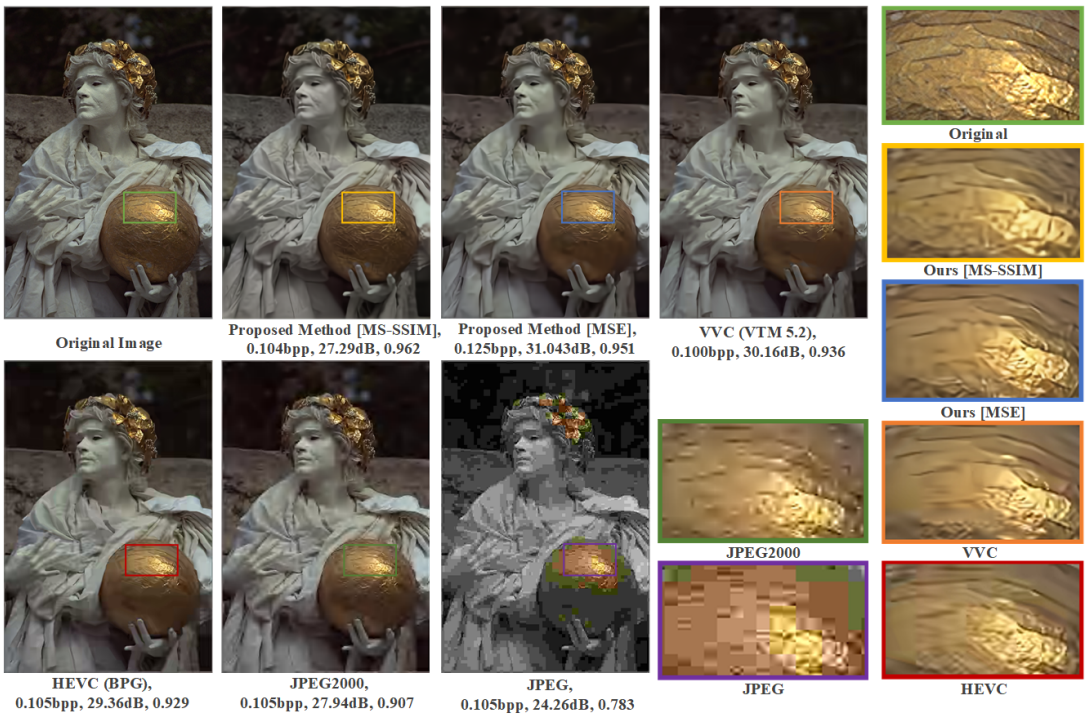


Figure 7: Visualization of reconstructed images *kodim17* from Kodak dataset.

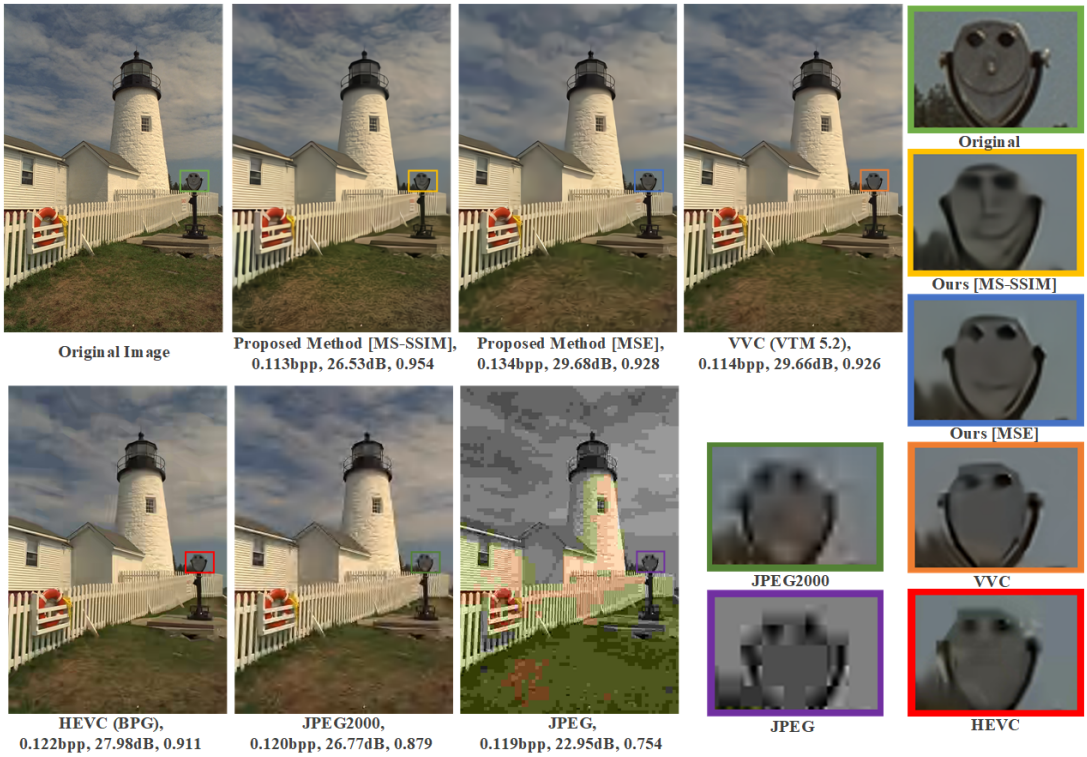


Figure 8: Visualization of reconstructed images *kodim19* from Kodak dataset.

Fig. 9 shows the reconstructed image *andrew-ruiz-376* with the resolution of 2048×1361 from CLIC professional validation dataset, with approximately 0.10 bpp and a compression ratio of 240:1. Due to the high resolution, the thumbnail seems to have very perceptually good quality. When zooming into the behind building, our method optimized by MS-SSIM has well preserved windows, but ghosting artifacts appear in traditional codecs, like VVC and HEVC. Blocking artifacts are quite visible in JPEG 2000 and JPEG compressed images.

Fig. 10 shows the reconstruction image *clem-onojeghuo-33741* with the resolution of 2048×1365 from CLIC professional validation dataset, with nearly 0.12 bpp and a compression ratio of 200:1. The water ripples are well-preserved only in our compressed images, but other compressed images look quite blurry. Besides, our method and VVC has not shown clear blocking artifacts, while blocking artifacts clearly appear in HEVC, JPEG2000 and JPEG compressed images.

Figure 9: Visualization of reconstructed images *andrew-ruiz-376* from CLIC validation dataset.

All the datasets are publicly available.

References

[1] Z. Cheng, H. Sun, M. Takeuchi, J. Katto, *"Deep Residual Learning for Image Compression"*, CVPR Workshop, pp. 1-4, June 16-20, 2019. 1

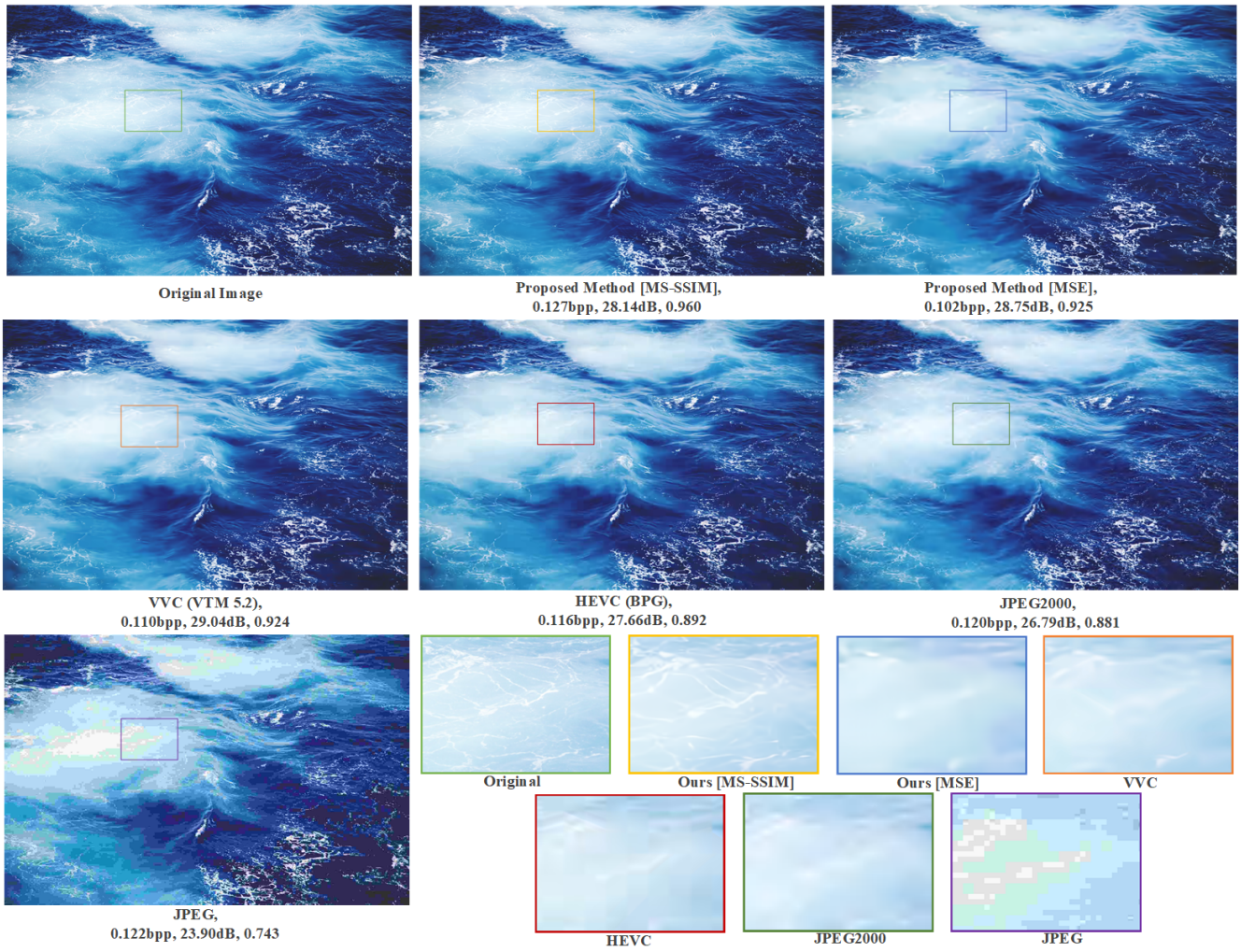


Figure 10: Visualization of reconstructed images *clem-onojeghuo-33741* from CLIC validation dataset.

[2] K. He, X. Zhang, S. Ren and J. Sun, *"Deep Residual Learning for Image Recognition"*, arXiv.1512.03385, 2015. 1

[3] Johannes Balle, D. Minnen, S. Singh, S. J. Hwang, N. Johnston, *"Variational Image Compression with a Hyperprior"*, Intl. Conf. on Learning Representations (ICLR), pp. 1-23, 2018. 1, 2

[4] D. Minnen, J. Ballé, G. Toderici, *"Joint Autoregressive and Hierarchical Priors for Learned Image Compression"*, arXiv.1809.02736. 1, 2

[5] J. Lee, S. Cho, S-K Beack, *"Context-Adaptive Entropy Model for End-to-end optimized Image Compression"*, Intl. Conf. on Learning Representations (ICLR) 2019. 1, 2

[6] G. Sullivan and S. Estrop, *"Recommended 8-bit YUV Formats for Video Rendering"*, 2018, online: https://docs.microsoft.com/en-us/windows/desktop/medfound/recommended-8-bit-yuv-formats-for-video-rendering 4

[7] Kodak Lossless True Color Image Suite, Download from http://r0k.us/graphics/kodak/ 6

[8] Workshop and Challenge on Learned Image Compression, CVPR2019, http://www.compression.cc/challenge/ 6