Supplementary Material

ELIC: Efficient Learned Image Compression with Unevenly Grouped Space-Channel Contextual Adaptive Coding

1. Detailed network architecture

1.1. Architecture of transform networks

Analyzer g_a	Synthesizer g_s
in: 3-channel image	in: M-channel symbols
Conv 5×5, s2, N	Attention
ResBottleneck$\times$3	TConv 5×5, s2, N
Conv 5×5, s2, N	ResBottleneck$\times$3
ResBottleneck$\times$3	TConv 5×5, s2, N
Attention	Attention
Conv 5×5, s2, N	ResBottleneck$\times$3
ResBottleneck$\times$3	TConv 5×5, s2, N
Conv 5×5, s2, M	ResBottleneck$\times$3
Attention	TConv 5×5, s2, 3

Table 1. Architecture of ELIC main transform networks.

Analyzer g_a	Synthesizer g_s
in: 3-channel image	in: M-channel symbols
Conv 5×5, s2, N	TConv 5×5, s2, N
ResBottleneck$\times$1	ResBottleneck$\times$1
Conv 5×5, s2, N	TConv 5×5, s2, N
ResBottleneck$\times$1	ResBottleneck$\times$1
Conv 5×5, s2, N	TConv 5×5, s2, N
ResBottleneck$\times$1	ResBottleneck$\times$1
Conv 5×5, s2, M	TConv 5×5, s2, 3

Table 2. Architecture of ELIC-sm main transform networks.

As shown in Table 1 and Table 2, our main networks are adapted from those previously used by Ballé *et al*. (2018), Minnen *et al*. (2018) and Minnen *et al*. (2020). We replace GDN/IGDN layers by residual blocks, as mentioned in the main text. We follow previous works to introduce a variable N representing the channel number of intermediate features, and lift the output channel number of analyzer to M. Thus, the coding-symbol $\hat{\boldsymbol{y}}$ is an $H \times W \times M$ tensor.

We adopt residual bottleneck structures as nonlinear transform, which have a dimensional bottleneck which can both provide larger receptive field and keep the volume of calculation acceptable. The channel number of intermediate features before and after 3×3 convolution is $\frac{N}{2}$. We sequentially stack 3 such residual blocks after each down/up-sampling convolution in ELIC (Table 1), and instead use only one at each position in ELIC-sm (Table 2). We also use the attention modules adopted by Cheng *et al*. in the larger ELIC models to further enhance the nonlinearity.

We simply adopt the three-layer hyper analyzer and synthesizer, following previous works (Minnen *et al*., 2018; Minnen *et al*., 2020; He *et al*., 2021). The hyper synthesizer output is an $H \times W \times (2M)$ tensor $\boldsymbol{\Psi}$ as shown in Figure 7 in the main text.

1.2. Architecture of SCCTX networks

Following Minnen *et al*. (2020), we use 5×5 convolutions to analyze cross-channel redundancy. The architecture of g_{ch} network is frankly sketched from Minnen *et al*. (2020). We follow the suggestions of Minnen *et al*. (2018) and He *et al*. (2021) and use single-layer 5×5 autoregressive/checkerboard-masked convolution as spatial context model g_{sp}. Both $g_{\text{sp}}^{(k)}$ and $g_{\text{ch}}^{(k)}$ output $H \times W \times (2M^{(k)})$ features, where $M^{(k)}$ is the channel number of the k-th channel group.

Therefore, we have the concatenated adaptive representation tensor $[\boldsymbol{\Phi}_{\text{sp},i}^{(k)}, \boldsymbol{\Phi}_{\text{ch}}^{(k)}, \boldsymbol{\Psi}] \in \mathbb{R}^{H \times W \times (4M^{(k)}+2M)}$. The parameter aggregation network linearly reduces the dimensions to $2M^{(k)}$.

2. Detailed experimental settings

We implement, train, and evaluate all learning-based models on PyTorch 1.8.1. We use NVIDIA TITANXP to test both RD performance and inference speed. To test the speeds, we reproduce previously proposed models and evaluate them under the same running conditions for fair comparison. Since most of the models adopt reparameterization techniques, we fix the reparameterized weights before testing the speed. We follow a common test protocol to test the latency with GPU synchronization. When testing each model, we drop the latency results (we do not drop them when evaluating RD performance) of the first 6 images to get rid of the influence of device warm-up, and average the

running time of remained images to get the precious speed results.

We do not enable the deterministic inference mode (*e.g.* `torch.backends.cudnn.deterministic`) when testing the model speeds for two reasons. First, we tend to believe that the deterministic issue can be well solved with engineering efforts, such as using integer-only inference. Thus, the deterministic floating-point inference is unnecessary. Second, the deterministic mode extremely slows down the speed of specific operators, like transposed convolutions which are adopted by ELIC and earlier baseline models (Ballé *et al*. and Minnen *et al*.), making the comparison somewhat unfair.

We obtain the RD results of prior works by asking the authors via emails or accessing released data directly.

2.1. Visualization explanation

We visualize the coding-symbols to investigate the information compaction property. The visualization results of the lighthouse image is shown in Figure 2 of the main text. The ℓ-th gray scale image is the rescaled magnitude $\boldsymbol{f}^{\ell}$ of the ℓ-th symbol channel $\hat{\boldsymbol{y}}^{(\ell)}$:

$$\boldsymbol{f}^{(\ell)} = \frac{|\hat{\boldsymbol{y}}^{(\ell)}|}{\max \hat{\boldsymbol{y}}}, \quad \ell = 1, 2 \ldots, M \tag{1}$$

Inspired by the concept of energy in the signal processing community, we introduce the term energy to describe the average square value of each symbol channel:

$$e^{(\ell)} = \frac{1}{HW} \sum_{\hat{y}_i^{(\ell)} \in \hat{\boldsymbol{y}}^{(\ell)}} \hat{y}_i^{(\ell)2} \tag{2}$$

In the main text, Figure 2 presents the channels with the largest energy values. Figure 3-left further shows the logarithmic energy $\log(e)$ of each channel.

3. More rate-distortion results

We also calculate the BD-rate over VVC when adopting MS-SSIM as the distortion metric. The results are shown in Table 3. Several baselines are skipped since the original authors do not provide the MS-SSIM results evaluated on the MSE-optimized models.

For completeness, we evaluate our ELIC model at larger BPP range (from 1.0 to 1.5 on Kodak). We train these models with $\lambda = \{0.08, 0.16\}$. The high-rate models still perform well. We draw the results on larger RD curves (Figure 1 for PSNR and Figure 2 for MS-SSIM) to more clearly show the superiority of the proposed approach. We also evaluate the models on CLIC-Professional and report Figure 3.

We further evaluate our model performance when optimizing for MS-SSIM, following prior works. Thus, the distortion term of the loss function is replaced by $1 - \text{MSSSIM}(\boldsymbol{x})$. We use $\lambda = \{3, 12, 40, 120\}$ to train such models, following Cheng *et al*. (2020). Figure 4 shows the results. On Kodak, when achieving the same MS-SSIM, ELIC optimized for MS-SSIM saves about half of the bitrate compared with VVC, which is encouraging.

Model	**BD-Rate (%) (PSNR)**	**BD-Rate (%) (MS-SSIM in dB)**
ELIC (ours)	**-7.88**	**-12.73**
ELIC-sm (ours)	-1.07	-5.59
Minnen2020	1.11	-
Cheng2020[P]	3.89	-7.19
Minnen2018[P]	20.00	4.23
Ballé2018	40.85	16.41
Gao2021	-10.94	-
Guo2021	-7.02	-
Wu2021 [43]	-5.71	-
Xie2021	-0.54	-7.82
Cheng2020	3.35	-3.17
Minnen2018	14.92	-0.68
VVC	0.00	0.00

Table 3. BD-rates over VVC for learned image compression models. The BD-Rate data is calculated from rate-distortion curves over data points with BPP < 1 on Kodak. We present results of two distortion metrics, PSNR and MS-SSIM in dB. Models marked with [P] adopt parallel checkerboard context model and [S] denotes serial context model.

4. Image reconstruction results

See Figures 5,6, we compare the reconstruction results of the proposed ELIC and Cheng *et al*. (2020), since their synthesis latency is close. We also present the decoded image of VTM12.1 for reference.

5. Progressive decoding

In the main text, we present progressive decoding results with thumbnail synthesizer and zero filling. In Figures 7,8, We further show progressive decoding results with full synthesizer and/or mean filling.

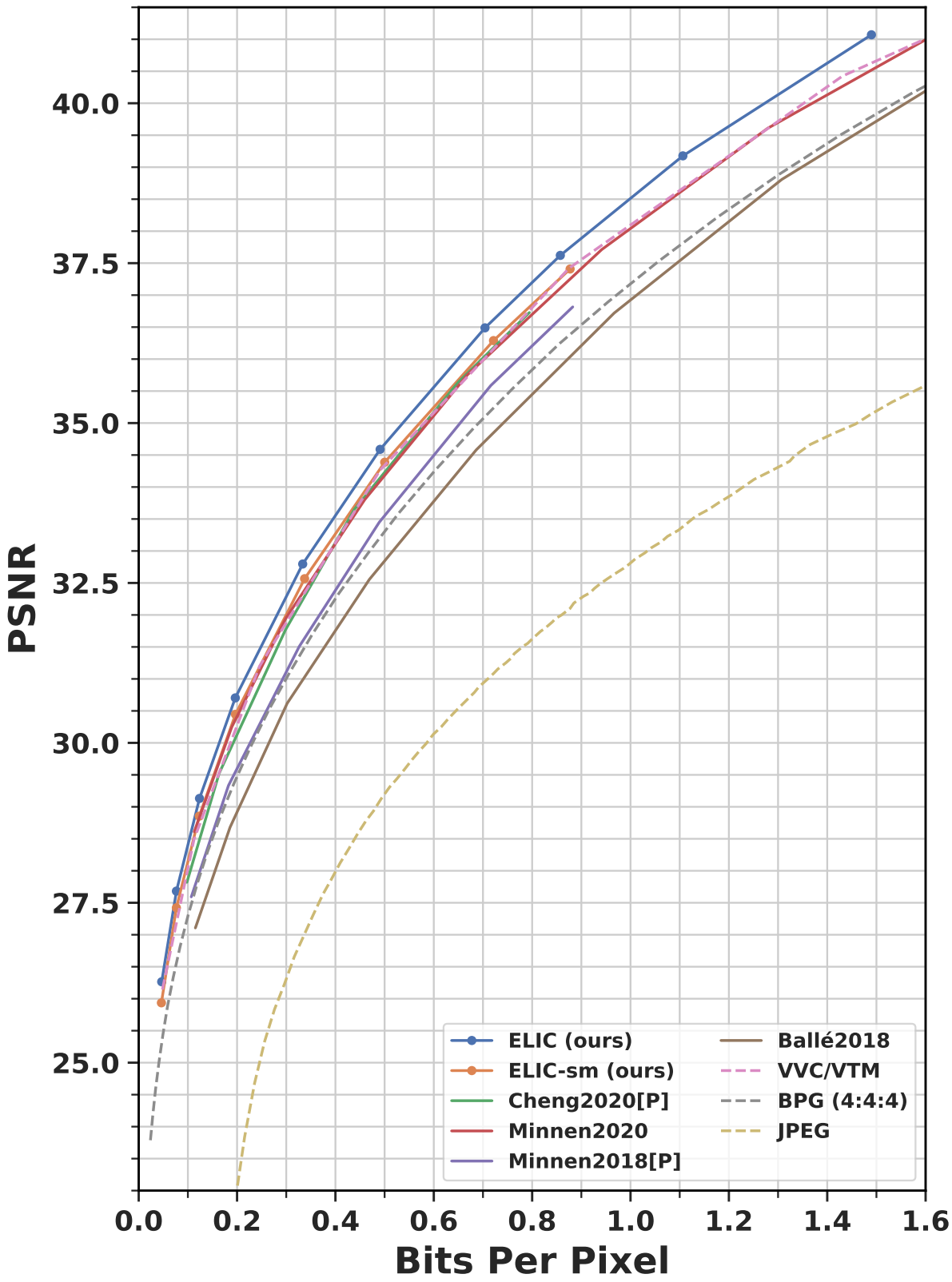

Figure 1. **PSNR-BPP curve on Kodak.** All models are optimized for minimizing MSE. All presented learning-based models can decode the 512×768 Kodak image in 100 microseconds.

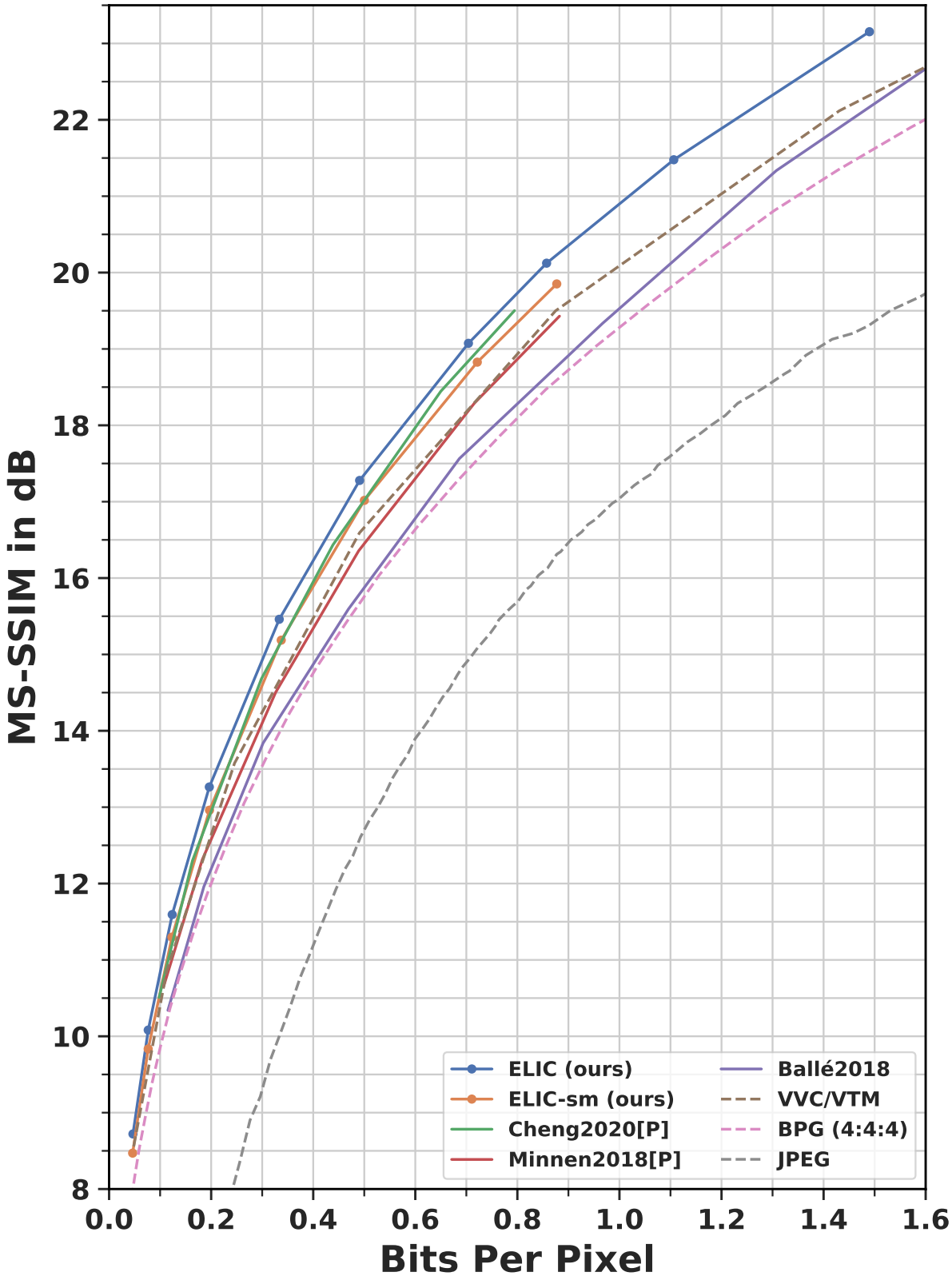

Figure 2. **MS-SSIM RD curve on Kodak.** All models are optimized for minimizing MSE. All presented learning-based models can decode the 512×768 Kodak image in 100 microseconds.

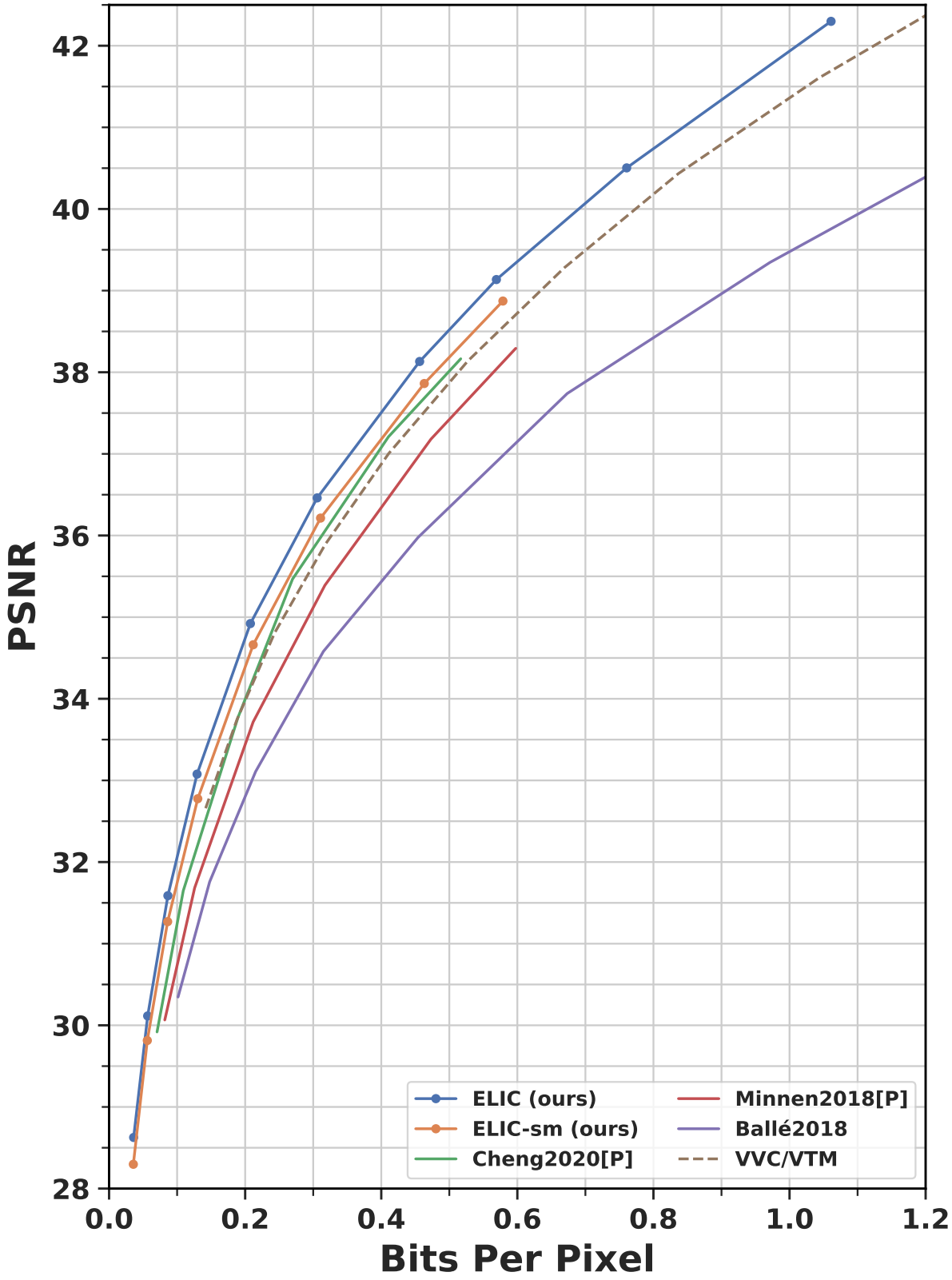

Figure 3. **PSNR-BPP curve on CLIC-Professional.** All models are optimized for minimizing MSE. All presented learning-based models can decode the 512×768 Kodak image in 100 microseconds.

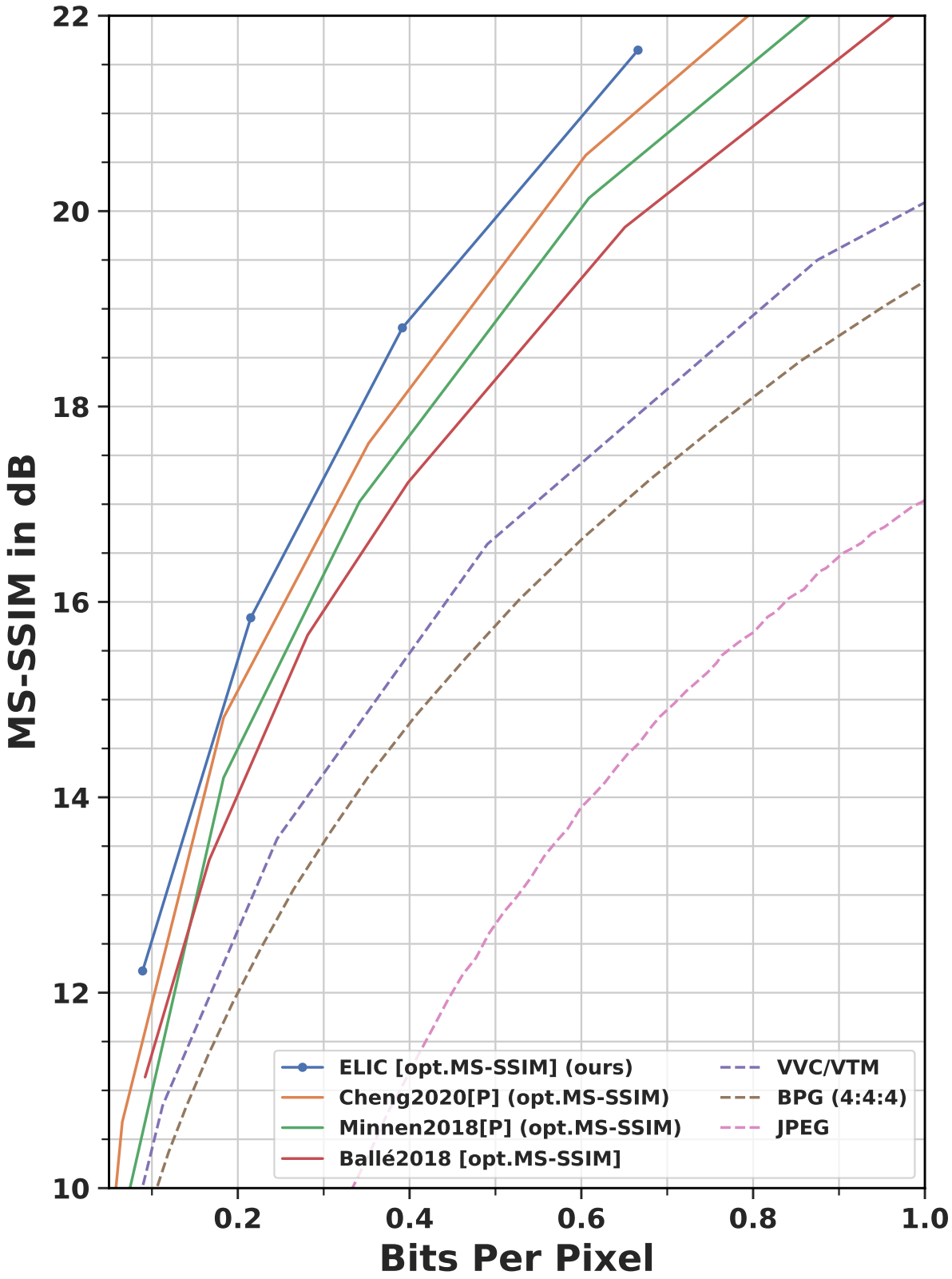

Figure 4. **MS-SSIM RD curve on Kodak (opt. MS-SSIM).** All models are optimized for maximizing MS-SSIM. All presented learning-based models can decode the 512×768 Kodak image in 100 microseconds.

(a) Cheng *et al*. (2020). BPP= 0.129. PSNR= 29.36

(b) Ours. BPP= 0.144. PSNR= 30.42

(c) Ground-truth.

(d) VTM12.1. BPP= 0.151. PSNR= 30.28

Figure 5. Qualitative comparison on reconstructed lighthouse (`kodim19`) image.

(a) Cheng *et al*. (2020). BPP= 0.155. PSNR= 29.85

(b) Ours. BPP= 0.127. PSNR= 31.66

(c) Ground-truth.

(d) VTM12.1. BPP= 0.151. PSNR= 31.65

Figure 6. Qualitative comparison on reconstructed sculpture (`kodim17`) image.

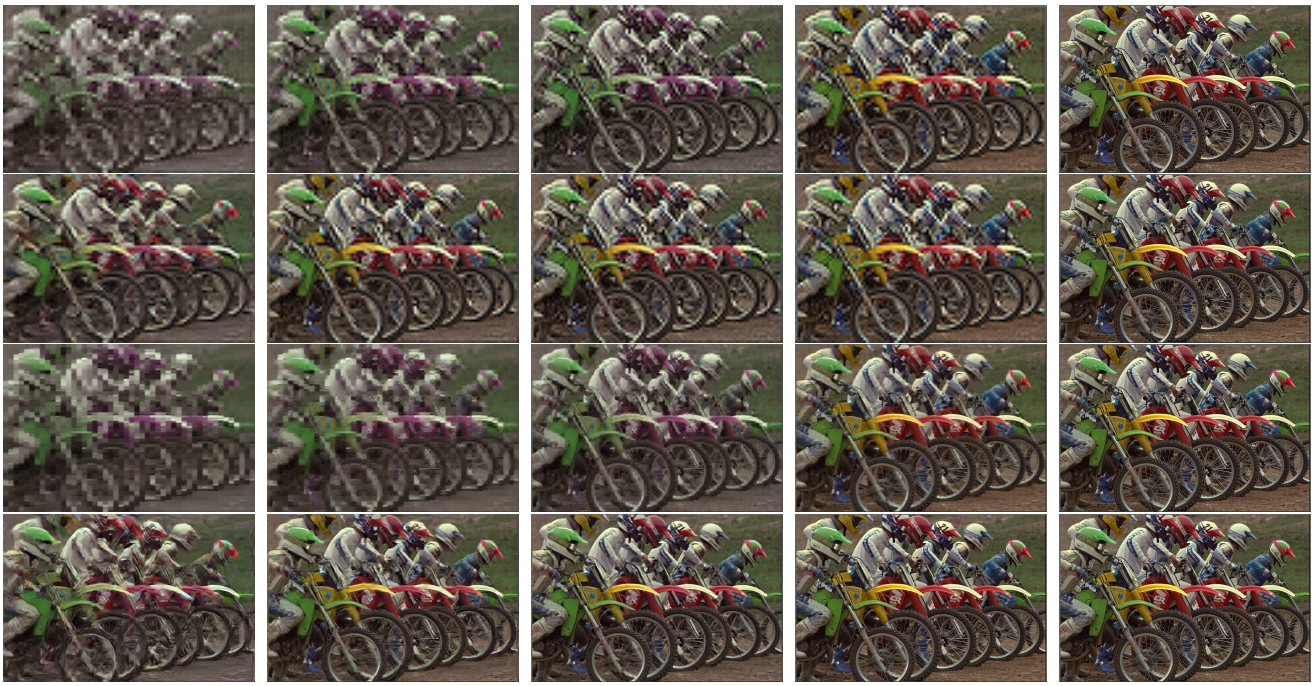

Figure 7. Progressive decoding results (kodim05).
Line 1 (thumb-zero): the first 4 groups are decoded with thumbnail decoder and zero filling (the same setting as the main text).
Line 2 (thumb-mean): the thumbnail-decoding results with mean filling.
Line 3 (full-zero): all the 5 groups are decoded with full synthesizer, using zero filling.
Line 4 (full-mean): the full-decoding results with mean filling.

Figure 8. Progressive decoding results of the house image (kodim24). The image order is the same as Figure 7.