Bridging the Gap between Gaussian Diffusion Models and Universal Quantization for Image Compression

Supplementary Material

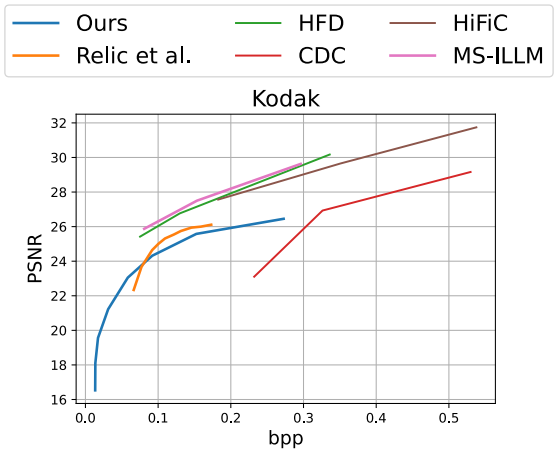

Figure 7. Rate-distortion comparison measured by PSNR on the Kodak dataset.

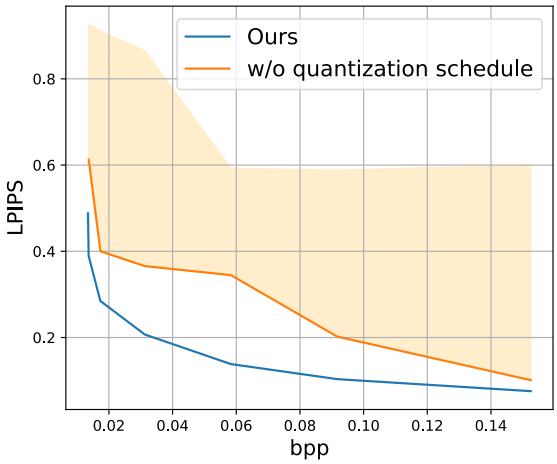

Figure 8. Ablation of our quantization schedule. The best performing combination of t_s and t_r is shown, as well as the range of all tested combinations (shaded).

7. Additional Results

7.1. Qualitative Results

Additional visualizations are shown in Figs. 10-12.

Furthermore, to provide examples of our method's ability to consistently produce realistic and plausible images, we show examples of images compressed over a range of bitrates in Fig. 13. Here it can be seen that at high bitrates, the reconstructions are accurate at a pixel level to the source image. As bitrate decreases, the images maintain high realism and semantic alignment and vary in low-level details (*e.g.*, bush, rock, or brick textures), rather than blurring artifacts traditionally seen in neural image compression.

7.2. Quantitative Results

In Fig. 7, we also include rate-distortion results measured in PSNR. We emphasize that for generative compression tasks, PSNR does not accurately reflect the true visual quality of image reconstructions. We include these results here solely for completeness.

7.3. Ablation of Quantization Schedule

While ablating the uniform diffusion model and universal quantization in our method is straightforward, ablating the quantization schedule is more arbitrary since we derive an objectively correct formulation. However, it can be ablated as follows.

The quantization schedule controls two parameters in our proposed pipeline: the SNR of $\hat{\mathbf{y}}_t$ and the number of denoising steps t to perform with the diffusion model at the receiver. The former can also be quantified as a function of t (as Δ_t is dependent on t; Eq. (7)); thus, we can consider two independent ts, one at the sender side (t_s), and one at the receiver (t_r). With our quantization schedule we find a closed form solution for $t_s = t_r$. Therefore, to ablate the quantization schedule, we can manually sweep over a range of t_s and t_r and compute image quality metrics for all possible combinations. Specifically, we choose all combinations of $t_s, t_r \in \{1, 2, 5, 10, 20, 30, 40\}$ except where $t_s = t_r$, as this corresponds to using our quantization schedule.

Results are shown in Fig. 8. As in the other ablations, we can see the quantization schedule has a significant impact on image quality, particularly at low bitrates.

7.4. Alternate Entropy Models

Our proposed method uses a relatively simple, yet standard mean-scale hyperprior entropy model [7, 25]. We also experimented with a newer, more complex entropy model (specifically Entroformer [30]) used in the baseline method of Relic *et al.* [31]. Quantitative results are shown in Fig. 9. It can be seen that the more powerful entropy model provides substantial rate-distortion performance gains in the higher bitrate range. However, it suffers in the lower rates.

Given the additional complexity of including this entropy model in our proposal, as well as a significantly larger

GPU memory requirement which prevents evaluation on high-resolution images, we elect to use the simpler mean-scale hyperprior entropy model in this work.

8. Image Generation with Uniform Diffusion Models

One concern when finetuning foundation diffusion models is catastrophic forgetting, where the model loses its ability to generate images. To validate our uniform diffusion model retains its image generation capability, we compare generated images of our model with Stable Diffusion v2.1 (the starting weights for our finetuning) using the same prompt, shown in Fig. 14. We perform text to image generation with DDIM sampling and generate images of 768^2 resolution. The seed used to sample the initial noise is the same within each pair. Prompts were arbitrarily generated by asking ChatGPT to write input prompts for Stable Diffusion with an emphasis on generating photorealistic images.

9. Derivation of Quantization Schedule

In this section we provide a derivation of our quantization schedule such that it matches the variance schedule of the original diffusion model (Eq. (10)). As described in Sec. 4.1 and Eq. (9), this is done by matching the SNR of the partially noisy data $\mathbf{y}_t$ of the original diffusion process with $\hat{\mathbf{y}}_t$ of our proposed forward process.

The standard Gaussian diffusion process is defined as:

$$\mathbf{y}_t = \sqrt{\alpha_t}\mathbf{y}_0 + \epsilon, \quad \epsilon \sim \mathcal{N}(0, (1-\alpha_t)\mathbf{I}). \tag{11}$$

In this context, the signal to noise ratio of $\mathbf{y}_t$ is:

$$\text{SNR}(\mathbf{y}_t) := \frac{\mathbb{E}[S^2]}{\text{Var}[N]} = \frac{\mathbb{E}[(\sqrt{\alpha_t}\mathbf{y}_0)^2]}{1-\alpha_t}. \tag{12}$$

Using our proposed forward noising process in Eq. 7 and deriving the SNR of $\hat{\mathbf{y}}_t$:

$$\hat{\mathbf{y}}_t = \lfloor \sqrt{\alpha_t}\mathbf{y}_0 - \mathbf{u} \rceil_{\Delta_t} + \mathbf{u}, \quad \mathbf{u} \sim \mathcal{U}[{}^{-\Delta_t}/_2, {}^{\Delta_t}/_2] \tag{13}$$

$$\stackrel{d}{=} \sqrt{\alpha_t}\mathbf{y}_0 + \mathbf{u} \tag{14}$$

$$\text{Var}[\mathbf{u}] = \frac{1}{12}\left(\frac{\Delta_t}{2} - \frac{-\Delta_t}{2}\right)^2 = \frac{1}{12}\Delta_t^2 \tag{15}$$

$$\therefore \quad \text{SNR}(\hat{\mathbf{y}}_t) = \frac{\mathbb{E}[(\sqrt{\alpha_t}\mathbf{y}_0)^2]}{\frac{1}{12}\Delta_t^2} \tag{16}$$

Therefore, to match the SNR between Gaussian diffusion and our proposed method:

$$\text{SNR}(\hat{\mathbf{y}}_t) = \text{SNR}(\mathbf{y}_t) \tag{17}$$

$$\frac{\mathbb{E}[(\sqrt{\alpha_t}\mathbf{y}_0)^2]}{\frac{1}{12}\Delta_t^2} = \frac{\mathbb{E}[(\sqrt{\alpha_t}\mathbf{y}_0)^2]}{1-\alpha_t} \tag{18}$$

$$\frac{1}{12}\Delta_t^2 = 1 - \alpha_t \tag{19}$$

$$\Delta_t = \sqrt{12(1-\alpha_t)}. \tag{20}$$

10. Entropy Coding with Variable Width Quantization Bins

Entropy coding our latent $\hat{\mathbf{z}}$ to bitstream requires a probability model over the set of symbols to be encoded – this is parameterized by the entropy model. As discussed by Ballé *et al*. [7, 8], in order for the entropy model to better match the true distribution of transmitted symbols, the underlying density model of the entropy model is convolved with a box filter of unit width, which lends itself to easy computation by evaluating the cumulative distribution (Eq. (29) in [8]):

$$p_{\hat{\mathbf{z}}}(\hat{\mathbf{z}}) = \Big(p * \mathcal{U}(-{}^1/_2, {}^1/_2)\Big)(\hat{\mathbf{z}}) \tag{21}$$

$$= c(\hat{\mathbf{z}} + {}^1/_2) - c(\hat{\mathbf{z}} - {}^1/_2) \tag{22}$$

where p is the underlying density model and c is the cumulative of p. Intuitively, to estimate the probability of some discretized value $\hat{\mathbf{z}}_i$ we integrate the density model over the range of unquantized values that result in $\hat{\mathbf{z}}_i$ after quantization.

However, an often overlooked fact is that the width of this box filter must equal the quantization bin width – using integer quantization requires a unit width box filter. Therefore, when quantizing with arbitrary bin width, as done in this paper, we must implement an entropy model which estimates probabilities according to this width:

$$p_{\hat{\mathbf{z}}}(\hat{\mathbf{z}}) = c(\hat{\mathbf{z}} + {}^\Delta/_2) - c(\hat{\mathbf{z}} - {}^\Delta/_2) \tag{23}$$

This does not lend itself to an intuitive implementation into current entropy modeling frameworks. However, we find that this issue can be resolved, along with a simple implementation of non-integer quantization, by scaling $\hat{\mathbf{z}}$ by Δ_t before quantization and entropy coding and rescaling back upon decoding:

$$\hat{\mathbf{z}} = \left\lfloor \frac{\sqrt{\alpha_t}\mathbf{y}}{\Delta_t} - \mathbf{u} \right\rceil, \quad \hat{\mathbf{y}}_t = (\hat{\mathbf{z}} + \mathbf{u}) \cdot \Delta_t, \tag{24}$$

where $\mathbf{u} \sim \mathcal{U}(-{}^1/_2, {}^1/_2)$ and $\Delta_t = \sqrt{12(1-\alpha_t)}$.

This is equivalent to Eq. (7) but allows the standard Eq. (22) to be used for probability estimation. This requires Δ_t to be known by the reciever, however incurs no rate penalty as Δ_t is derived from t which is already transmitted as side information.

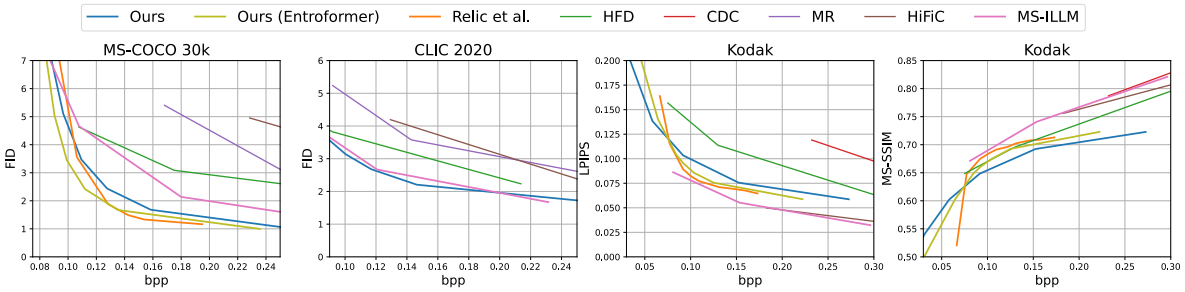

Figure 9. Additional rate-distortion comparisons including our method with an alternate entropy model.

11. Reproduction of Baselines

In the following paragraphs we detail specifics on how image reconstructiond and quantitative metrics were obtained for each baseline.

Relic *et al.* Reconstructions and quantitative metrics were obtained through personal communication.

HFD. Reconstructions were obtained through personal communication, also available at `http://theis.io/hifidiff//`. Quantitative metrics are used as reported in their paper.

MR. Quantitative metrics are used as reported in their paper.

HiFiC. Pretrained models are available in Tensorflow Compression `tfci` as `hific-hi`, `hific-mi`, and `hific-lo`.

CDC. We produce images on our evaluation datasets using the official code release at `https://github.com/buggyyang/CDC_compression` (commit `742de7f`). We take the epsilon parameterized models trained with L_1 loss and axuillary LPIPS perceptual loss with weighting parameter $\rho = 0.9$. During inference, we use 1000 sampling steps.

VTM We use VTM 19.2, available at `https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/releases/VTM-19.2` and run with the following commands:

```
# Encode
    EncoderAppStatic
    -c encoder_intra_vtm.cfg
    -i $INPUT
    -q $QP
    -o /dev/null
    -b $OUTPUT
    -wdt $WIDTH
    -hgt $HEIGHT
    -fr 1
    -f 1
    --InputChromaFormat=444
    --InputBitDepth=8
    --ConformanceWindowMode=1
    --InputColourSpaceConvert=RGBtoGBR
    --SNRInternalColourSpace=1
    --OutputInternalColourScace=0

# Decode
    DecoderAppStatic
    -b $OUTPUT
    -o $RECON
    -d 8
    --OutputColourSpaceConvert=GBRtoRGB
```

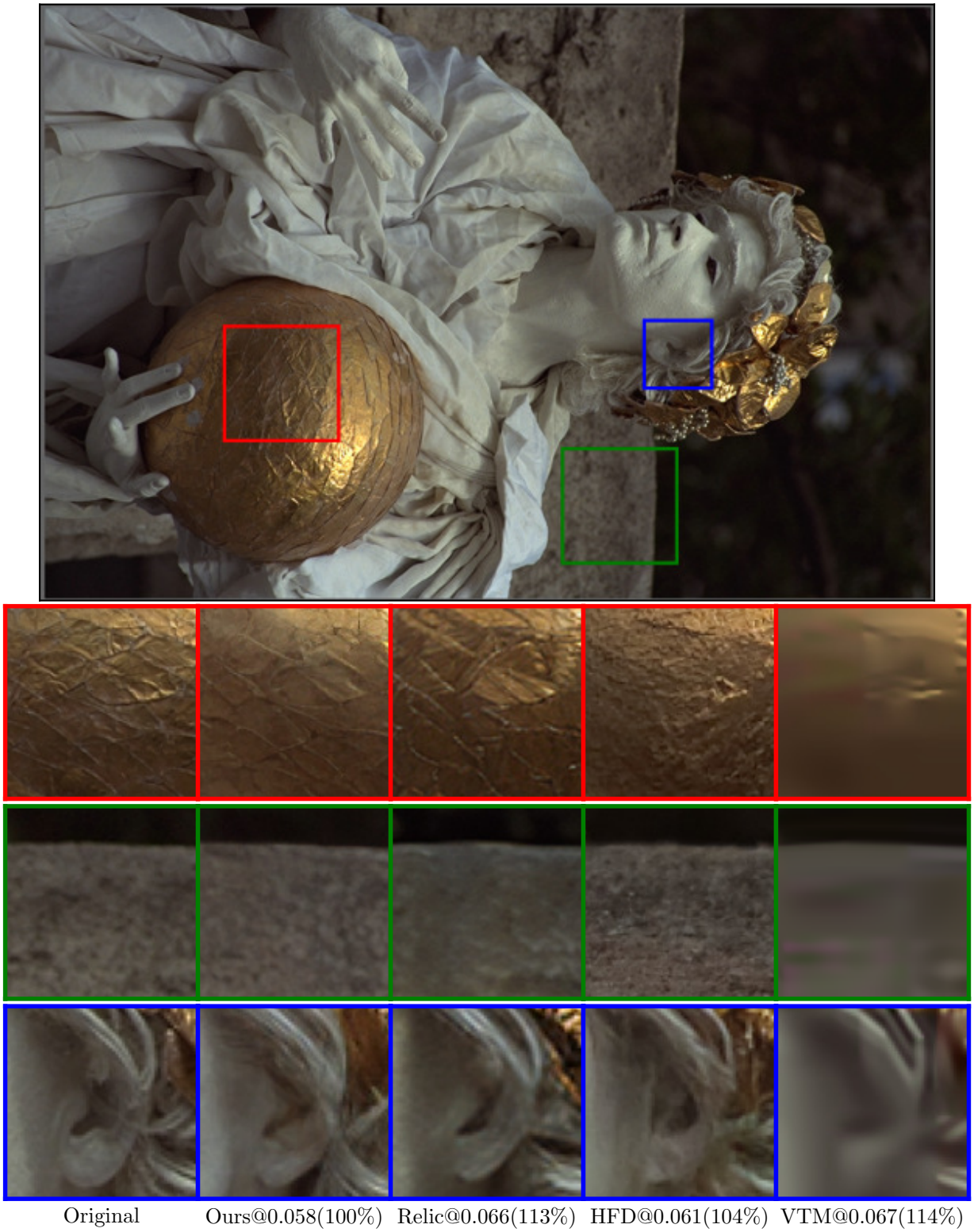

Figure 10. Additional visual comparisons on *kodim17* between our method and Relic *et al*. [31], HFD [19], and VTM [1]. Images are labeled as [Method]@bpp and additionally shown as a percentage of our method. Here Relic *et al*. flattens textures, and HFD incorrectly synthesizes content or oversharpens the image.

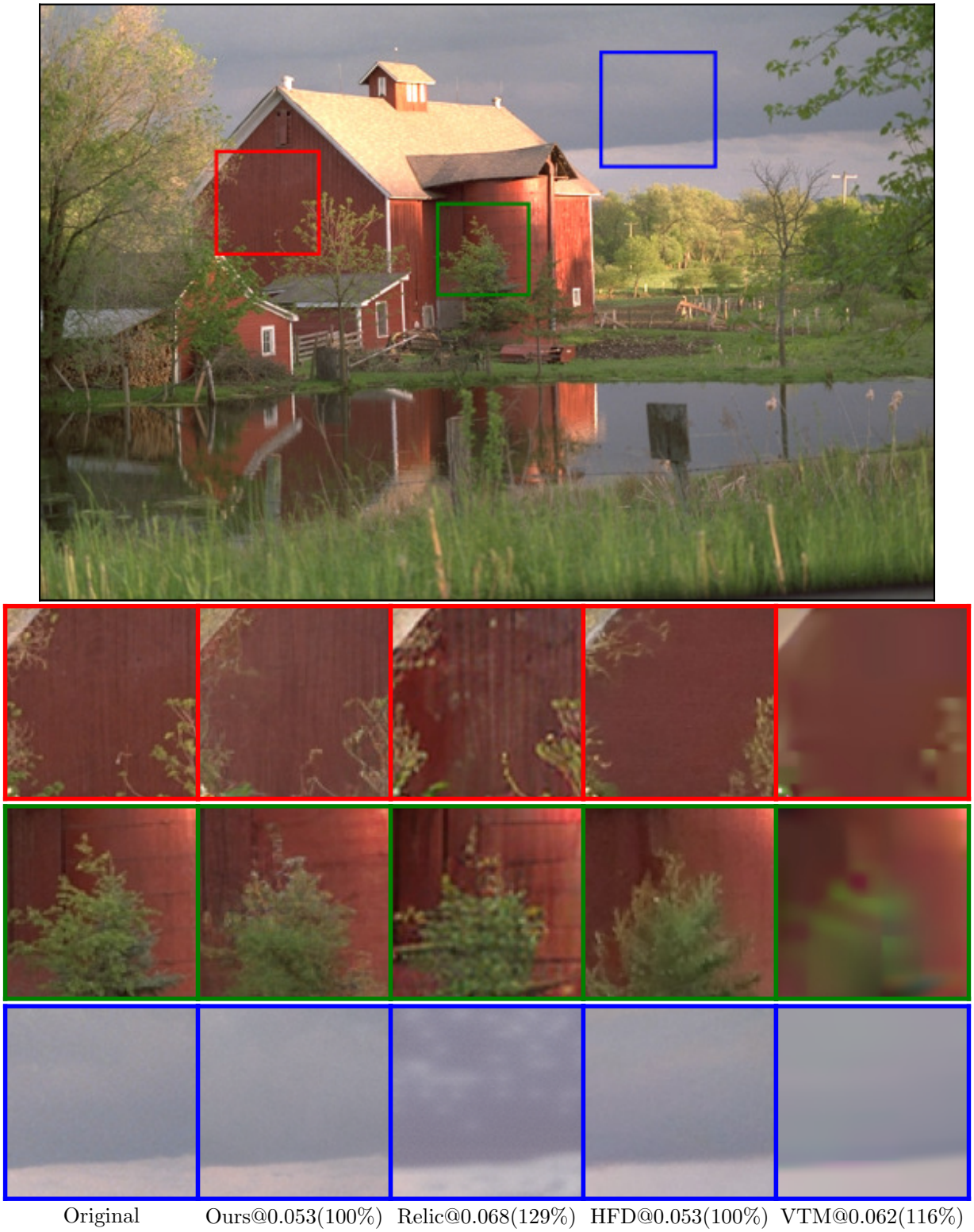

Figure 11. Additional visual comparisons on *kodim22* between our method and Relic *et al*. [31], HFD [19], and VTM [1]. Images are labeled as [Method]@bpp and additionally shown as a percentage of our method. Here Relic *et al*. introduces color artifacts and oversaturation, and HFD is not as detailed.

Figure 12. Additional visual comparisons on *kodim07* between our method and Relic *et al*. [31], HFD [19], and VTM [1]. Images are labeled as [Method]@bpp and additionally shown as a percentage of our method. Here Relic *et al*. and HFD do not correctly synthesize fine details or textures.

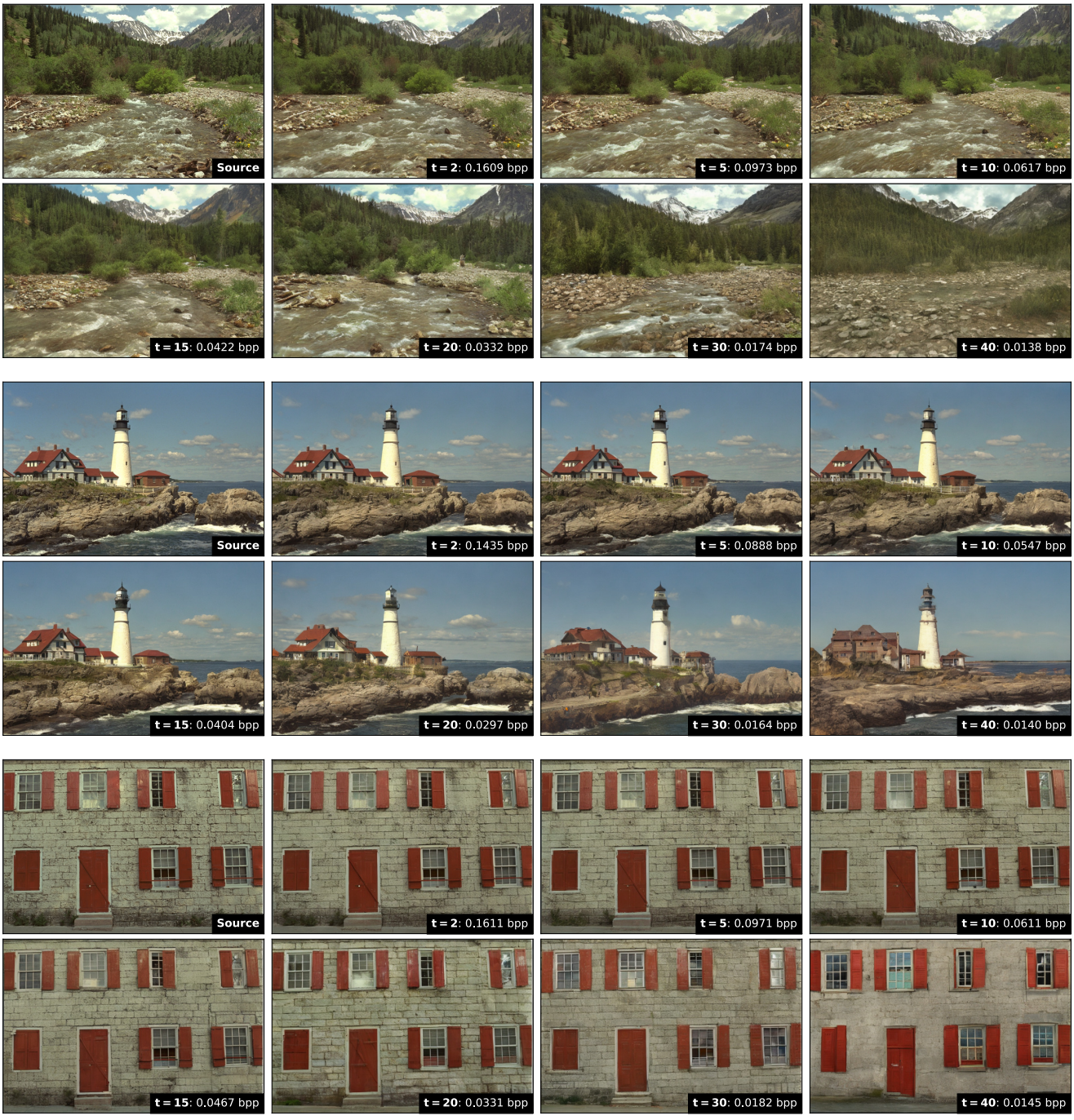

Figure 13. Visual example of image reconstructions produced over a range of bitrates. At high bitrate, our method produces reconstructions with high fidelity to the source image. As bitrate decreases, the images maintain a high realism and semantic alignment, tending towards differences in the spatial alignment of content. Best viewed digitally.

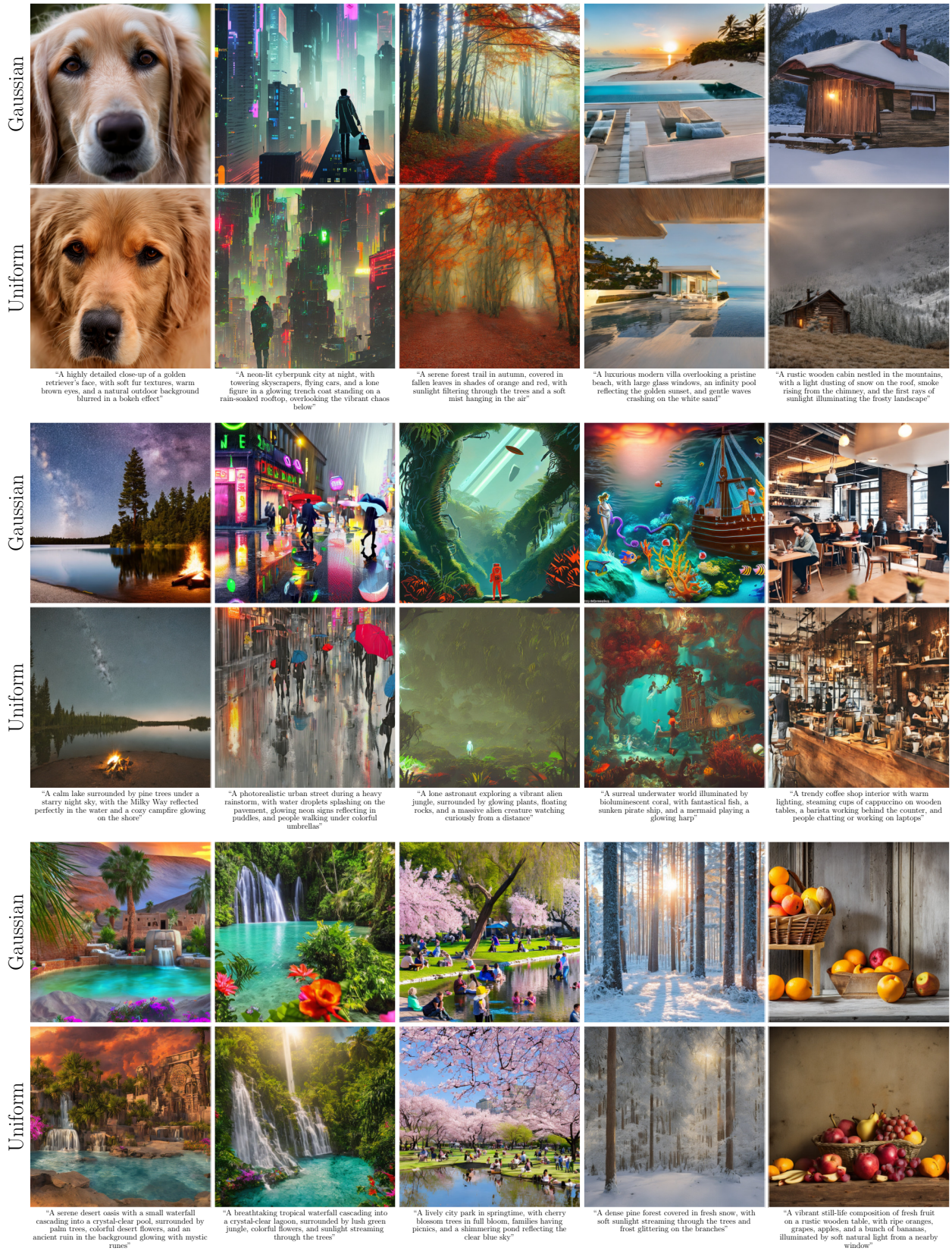

Figure 14. Image generation results between the standard Gaussian diffusion model versus our diffusion model finetuned for uniform noise. The prompt under each pair was arbitrarily generated using ChatGPT.