Rethinking Latent Variable in Learned Image Compression

Supplementary Material

A. Detailed Data-flow Diagram

Our network largely follow the architecture proposed in [27], and we just briefly repeat it in Fig. 7.

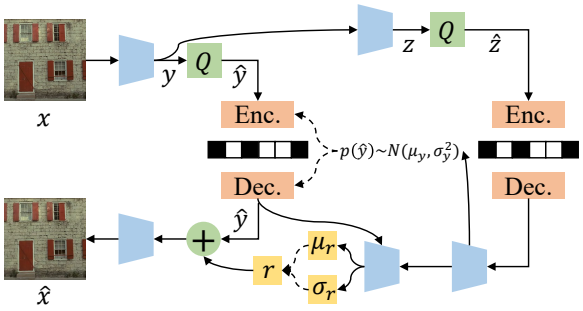


Figure 7. Data-flow diagram of our compression model with learned parameter compensation (LPC). Blue blocks denote image encoder/decoder and hyper encoder/decoder. Green blocks with Q denote quantization operation and green block with + is plus operation. Orange Blocks are arithmetic encoder/decoder. Yellow block with μ_r and σ_r are learned parameters for compensation distribution, these variables in Sec. 3.3 is denoted as μ_{learn} and σ_{learn}. The dashed lines in the figure represent the modeling of probability distributions.

B. More Intuition

B.1. Delve into Training

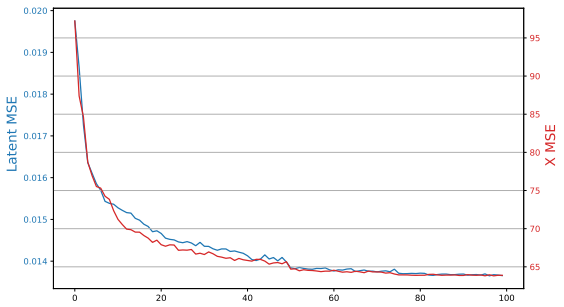


Figure 8. Comparison of loss trends for Mbt-mean [27] model with $\lambda = 0.0035$. The red line denotes Image level MSE and blue line is latent level MSE.

We observed that the latent-quantized MSE exhibits a similar trend to the image reconstruction MSE, as shown in Fig. 8. This empirical correlation strongly supports our claim: reducing quantization-induced latent error contributes to reducing image reconstruction error, thereby benefiting RD performance indirectly. Xie et al. [37] adopt an invertible encoder-decoder architecture, in which the latent representations $\mathbf{y}$ and $\hat{\mathbf{y}}$ can be interpreted as deeper-layer counterparts of the original image $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$.

B.2. Decompose Reconstruction Error

From an information flow perspective, $\mathbf{y}$ is designed to efficiently encode the macroscopic structure and primary semantic content of the image. It operates akin to an information bottleneck, prioritizing the most perceptually salient features at low bitrate [27, 32]. While $\mathbf{y}$ enables discernible image content, the visual quality and realism of an image heavily depend on its high-frequency details, textures, and subtle semantic cues.

Based on this analysis and assumption, we analyzed the causes of image reconstruction errors from both the pixel domain and the frequency domain. We define three error metrics MSE_t, MSE_a and MSE_q in Eq. (11):

$$\begin{aligned} \text{MSE}_\text{t} &= \text{MSE}(X, X_\text{q}) \\ \text{MSE}_\text{a} &= \text{MSE}(X, X_\text{uq}) \\ \text{MSE}_\text{q} &= \text{MSE}(X_\text{uq}, X_\text{q}) \end{aligned} \tag{11}$$

where X is the raw image, X_q is the reconstructed image with latent variable quantization, and X_uq denotes the reconstructed image without latent variable quantization. We consider MSE_t as the overall image reconstruction error; MSE_a as the error introduced by the model network(or, more precisely, attributed to the compression process); and MSE_q as the error resulting from quantization.

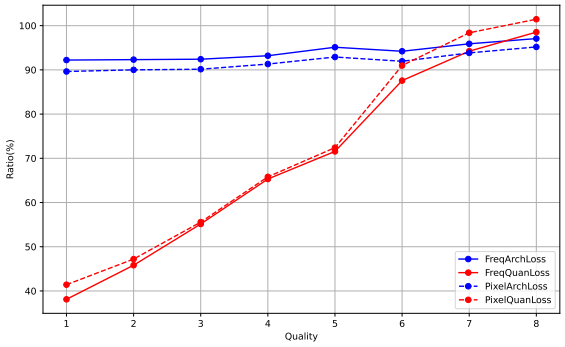


Figure 9. Performance Comparison of Different Loss Functions Across Varying Reconstruction Qualities. $Freq$ represents frequency domain analysis while $Pixel$ denotes pixel domain.

To better illustrate the trend of loss variation, we transformed the comparison between MSE_a and MSE_q into a

comparison between $\frac{\text{MSE}_a}{\text{MSE}_t}$ and $\frac{\text{MSE}_q}{\text{MSE}_t}$. As shown in Fig. 9, The "ArchLoss"-related metrics, namely FreqArchLoss and PixelArchLoss, appear to be insensitive to changes in quality, consistently maintaining a high Ratio value. Conversely, the "QuanLoss"-related metrics, including FreqQuanLoss and PixelQuanLoss, show a positive correlation with quality: higher quality leads to a higher Ratio value, clearly indicating their high sensitivity to changes in quality.

ArchLoss shows minor variations with changes in reconstruction quality, whereas QuanLoss exhibits more significant fluctuations. We consider ArchLoss as the error introduced by the model itself; consequently, ArchLoss remains relatively stable as long as the model architecture is not altered. Conversely, QuanLoss, being an error caused by quantization, should be reduced.

Thus, to recover $\mathbf{y}$ from $\hat{\mathbf{y}}$ is not merely a simple error correction. It is a critical step that leverages the hierarchical nature of latent variable information, bridges the gap between model training and inference, and achieves an efficient rate-distortion balance.

C. More Experiment

C.1. Dataset Capacity Expansion and Batch Size

In Sec. 4.5, we mentioned that Inverse training leads to an increase in dataset size, which can consequently extend the training duration. In this section, we include additional experiments regarding this phenomenon.

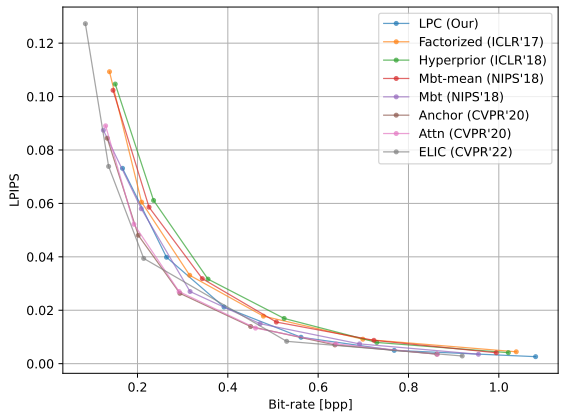
Inverse	Size	Eval Loss↓	Time per Epoch (min)
✓	16	0.7262	23.2
	8	0.7227	44.0
✗	16	0.7313	12.2
	8	0.7260	24.0

Table 5. Dataset capacity expansion and batch size comparison.

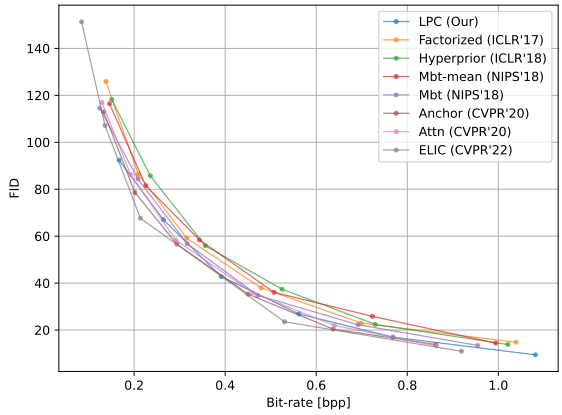
In Tab. 5, we test the relationship between enabling Inverse Training and Batch Size. As we well know, increasing the batch size typically leads to a certain degree of model performance degradation. The data in Tab. 5 also corroborates this point. However, we can observe that with Inverse training enabled, the performance degradation caused by increasing the batch size is smaller. Additionally, with Inverse training enabled, the dataset size doubles. Training with Inverse on and batch size is 16 should shares same iteration with training with Inverse off and batch size is 8. It is interesting to see that the limited reduction in model performance coupled with a certain degree of training acceleration. We believe this acceleration is due to the reduction in I/O operations during training.

C.2. More Evaluation Details

In this section, we primarily supplement the LPIPS and FID performance of the LPC module on the Kodak dataset.



(a) LPIPS



(b) FID

Figure 10. Generative Model Evaluation Metrics on Kodak Dataset

For LPIPS in Fig. 10a, at the highest bit-rates, LPC not only outperforms Mbt-mean but also surpasses other more complex models like Anchor [10], Attn [10], and ELIC [14]. This indicates that LPC is competitive in achieving excellent perceptual quality, especially when higher bit-rates are available.

For FID in Fig. 10b, LPC also outperforms some other more complex models in terms of FID, such as Anchor [10] and Attn [10]. This highlights LPC's advantage in achieving high image fidelity, particularly at higher bit-rates.

C.3. Qualitative Results for CLIC Datasets

In this section, we primarily focus on LPC qualitative performance on two CLIC datasets. As shown in Tab. 6, Our LPC module demonstrates similarly significant performance improvements on the CLIC.p. and CLIC.m. datasets as it did on the Kodak dataset.

Dataset	Method	BD-Rate(%)↓	BD-PSNR(dB)↑
CLIC.m.	Mbt-mean	0.0	0.0
	LPC	**-6.38**	**+0.26**
CLIC.p.	Mbt-mean	0.0	0.0
	LPC	**-5.97**	**+0.24**

Table 6. Quanlitive Comparison on CLIC.m. and CLIC.p. dataset.

C.4. Comparison with other Works

In this section, We compare the RD performance with other quantization modeling works, including Gumbel Annealing [38] and Latent Residual Prediction [26]. All comparison results are based on Kodak dataset.

Gumbel Annealing(SGA). We use PyTorch implementation [12] of SGA [38] and test it with Hyperprior [5]. As show in Tab. 7, SGA method indeed brings significant performance improvements, but the additional time overhead it introduced is also evident. Furthermore, our FPC module primarily serves as a preliminary exploration for the LPC module. We believe that the LPC module can also bring significant performance improvements when applied to the Hyperprior architecture. Additionally, FPC and LPC, as embedded modules, participate in the model training process, whereas the SGA method is more akin to a post-processing technique that primarily operates during the inference stage.

Method	Enc.&Dec.(s)	BD-Rate(%)↓
Hyperprior	0.22	0.0
FPC	0.22	+0.79
SGA	53.87	-8.8%

Table 7. Quanlitive Comparison on with SGA [38]

Latent Residual Prediction(LRP). We reimplement LRP with PyTorch. Please note that for a fair comparison of the two residual compensation methods, we did not implement the channel conditioning mentioned in the original paper [26]. As shown in Tab. 8, our LPC module achieves better RD-performance.

Method	BD-Rate(%)↓	BD-PSNR(dB)↑
Mbt-mean	0.0	0.0
LPC	-5.54	+0.26
LRP	-4.59	+0.20

Table 8. Quanlitive Comparison on with LRP [26]

We believe that LRP and LPC in fact represent the evolutionary process of Autoencoder paradigm. Both LRP and LPC are dedicated to reconstruct quantization residuals from quantized latent variables. While LRP attempts to directly establish its residuals, LPC tends to establish the probability distribution of the residuals. Considering the differences between Autoencoders[4] and Variational Autoencoders[18], and their disparity in generalization performance, it is understandable that LPC can achieve better performance than LRP.