Supplementary Materials

S1. Data Compression Technologies

Data compression reduces the size of digital data, without compromising the essential information they contain. This is achieved by identifying and eliminating redundancies within the data, resulting in a more compact representation that requires less storage space or bandwidth for transmission.

The three key techniques widely used in multimedia compression are prediction, transformation, and entropy coding.

Prediction-based compression leverages the inherent patterns and correlations within the data to predict future or missing elements. In audio compression, future audio frames can be predicted based on previous frames, as seen in techniques like linear predictive coding (LPC) [1]. Video compression often employs prediction, both within frames (intra-frame prediction) and between frames (inter-frame prediction), as demonstrated in the H.264 video codec [2].

Transformation-based compression aims to convert the data representation from a higher-dimensional space to a lower-dimensional space, reducing the overall data size. In audio, image, and video compression, techniques like the Discrete Cosine Transform (DCT) [3] are widely used to transform the digital media into the frequency domain, where high-frequency components can be selectively discarded with minimal impact on visual quality. The transformed data can then be quantized and encoded using entropy coding methods.

Entropy coding is a lossless compression technique that eliminates statistical redundancies in the data by assigning shorter codes to more frequent symbols and longer codes to less frequent symbols. Two most commonly used entropy coding techniques are Huffman coding [4] and arithmetic coding [5]. Huffman coding is a variable-length coding method that constructs a prefix-free code to minimize the average code length, while arithmetic coding can achieve higher compression ratios by representing the entire input sequence as a single code.

S2. Arithmetic coding for compression

Our approach uses arithmetic coding [6] for data compression, so we introduce it in more detail to make the paper self-contained.

S2.1. A brief introduction to arithmetic coding

Arithmetic coding aims to map a sequence over a finite alphabet Σ into a value in the interval $[0, 1)$. Each sequence $\omega = \omega_1\omega_2\cdots\omega_n \in \Sigma^n$ is associated with a probability mass function p_n that satisfies $p_n(\omega) = \sum_{x\in\Sigma} p_{n+1}(\omega x)$. The arithmetic encoder compresses the sequence symbol by symbol while iteratively narrowing down the interval $I_0 = [0, 1)$.

Specifically, for any $1 \le k \le n$, suppose $I_{k-1} = [a_{k-1}, b_{k-1})$ is the interval obtained after encoding ω_{k-1}. To encode ω_k, we define the interval $I_k = [a_k, b_k)$ as follows:

$$a_k = a_{k-1} + (b_{k-1} - a_{k-1}) \sum_{x\in\Sigma, x\prec\omega_k} p(x|\omega_{1:(k-1)}), \tag{1}$$

$$b_k = a_{k-1} + (b_{k-1} - a_{k-1}) \sum_{x\in\Sigma, x\preceq\omega_k} p(x|\omega_{1:(k-1)}), \tag{2}$$

where

$$p(x|\omega_{1:(k-1)}) = \frac{p_k(\omega_{1:(k-1)}x)}{p_{k-1}(\omega_{1:(k-1)})}$$

Finally, we select a value $\lambda \in I_n$ that has the shortest binary representation and use it as the arithmetic code for ω. Note that the length $l(\lambda)$ of the binary representation of λ is

$$\begin{aligned} l(\lambda) &\le \lceil -\log_2(b_n - a_n)\rceil + 1 \\ &= \left\lceil -\log_2 \prod_{i=1}^{n} p(\omega_i|\omega_{1:(k-1)}) \right\rceil + 1 \\ &= \lceil -\log_2 p_n(\omega)\rceil + 1. \end{aligned}$$

On the other hand, given λ, we can reconstruct the sequence ω in a similar manner. Starting with the interval $I_0 = [0, 1)$, assume that we have recovered $\omega_{1:(k-1)}$ and obtained the interval $I_{k-1} = [a_{k-1}, b_{k-1})$ that contains λ. The kth element of ω is the unique symbol $\omega_k \in \Sigma$ for which $I_k = [a_k, b_k)$ contains λ, where a_k and b_k are defined as shown in Formulas (1) and (2). This way, the original sequence ω can be exactly reconstructed from its arithmetic code λ, thus making arithmetic coding a lossless compression method.

S2.2. Arithmetic coding with unknown posterior distribution

A common scenario in practice is that the posterior distribution p is not known. In order to apply arithmetic coding, we have to find an approximate distribution q that is close to p. This is a long-time challenge dating back to 1960's when the well-known Solomonoff induction was proposed [7].

Solomonoff induction approximates the ground-truth distribution q with theoretical guarantee, but it relies on on the uncomputable semimeasure Solomonoff prior [8]. Recent years has witnessed many generative large models, for example, GPT series, LLaMA, and Chinchilla, which essentially estimate the ground truth distribution p

from a huge corpus. It is known [9, 10] that, if fully trained with log-loss function, the estimated distribution q converges to the ground truth probability p. Hence, we can bypass uncomputability of Solomonoff prior via large models.

In arithmetic coding, we use q instead of p and hence need $\lceil -\log_2 q_n(\omega) \rceil + 1$ bits to represent a sequence $\omega \in \Sigma^n$. Therefore, the expected number of bits needed to compress a sequence in Σ^n is

$$\begin{aligned} &\mathbb{E}_{\omega \sim p_n} \lceil -\log_2 q_n(\omega) \rceil + 1 \\ \leq &\mathbb{E}_{\omega \sim p_n} [-\log_2 q_n(\omega)] + 2 \\ = &H(p_n, q_n) + 2. \end{aligned}$$

Here, $H(p_n, q_n)$ is the cross-entropy between the true distribution and the estimated distribution. According to the theory of cross-entropy, the better p_n is approximated, the better the compression is, and the shortest possible length of the arithmetic code is 2 plus the entropy of p_n.

S3. Lossy Vedio Compression

The core idea of our compression approach is to leverage the capabilities of a generative model to compensate for the distortions introduced by a traditional video compression framework. The generative model's ability to "recall" or restore an image from an incomplete or damaged state, similar to the inpainting process, is key. The stronger a model's recall ability, the more diverse the range of natural image distributions it can cover.

Figure 1 shows the architecture of our method, which combines a Video Compression Framework (such as the DCVC series) based on an autoencoder, a diffusion U-net model, and an inverse strategy.

The encoding process is straightforward, and we can use any existing encoder from a transform-based video compression method. During decoding, the reference frame is first obtained from the Video Compression Framework, which suffers from average distortion due to quantization error. Then, at each denoising step in the diffusion model, the gradient of the squared error between the reference frame and the intermediate results is required to approximate the manifold of the source frame. The scale parameter ζ controls the update step size, where a larger ζ can produce results more similar to the source frame but may also lead to artifacts due to the model's inability to precisely reconstruct the source frame's manifold. Finally, we obtain a reconstructed frame that is perceptually consistent with human vision and similar to the source frame in terms of semantics and structure.

The key point is that the gradient of the Video Compression Framework is zero because of the presence of quantization. To address this, we employ a strategic approach where quantization is used during the encoding and decoding of the source frame, but additive noise is used instead of quantization at the denoising steps. Intuitively, since the errors generated during the quantization process are similar to adding noise, and the diffusion model is fundamentally a denoising process, it can

"remove" these errors and produce a result that is a trade-off between the source frame distribution and the natural image distribution the model has learned.

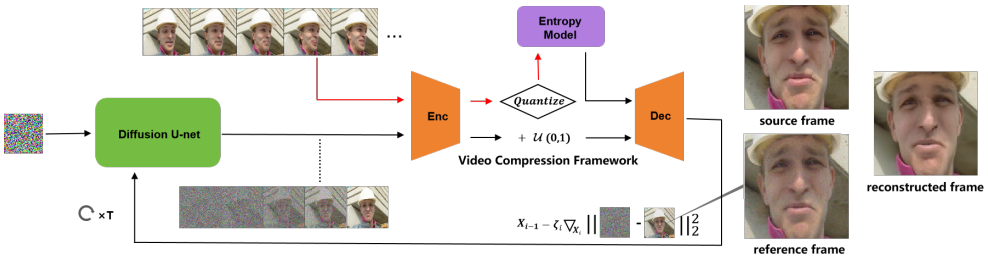

Fig. 1 Overview of LMCompress lossy video compression. The red lines represent the encoding process and the black lines represent the decoding process. The reference frame is obtained by Video Compression Framework. During the iteration of diffusion model, intermediate image is updated though gradient decent (uniform noise instead of quantization) with a scale parameter ζ and we get reconstructed frame after T rounds of iterations.

References

[1] Auristin, F.N., Mali, S.D.: Advanced audio compression for lossless audio coding using ieee 1857.2. International Journal Of Engineering And Computer Science (2016)

[2] Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the h.264/avc video coding standard. IEEE Transactions on Circuits and Systems for Video Technology (2003)

[3] Ahmed, N., Natarajan, T., Rao, K.R.: Discrete cosine transform. IEEE Transactions on Computers **C-23**(1), 90–93 (1974) https://doi.org/10.1109/T-C.1974.223784

[4] Huffman, D.A.: A method for the construction of minimum-redundancy codes. Proceedings of the IRE **40**(9), 1098–1101 (1952) https://doi.org/10.1109/JRPROC.1952.273898

[5] Pasco, R.C.: Source coding algorithms for fast data compression. PhD thesis, Stanford University CA (1976)

[6] Rissanen, J.J.: Generalized kraft inequality and arithmetic coding. IBM Journal of research and development **20**(3), 198–203 (1976)

[7] Solomonoff, R.: A formal theory of inductive inference. Inform. control **7**(1) (1964)

[8] Li, M., Vitanyi, P.: An Introduction to Kolmogorov Complexity and Its Applications. Springer (2019)

[9] Ortega, P.A., Wang, J.X., Rowland, M., Genewein, T., Kurth-Nelson, Z., Pascanu, R., Heess, N.M.O., Veness, J., Pritzel, A., Sprechmann, P., Jayakumar, S.M., McGrath, T., Miller, K.J., Azar, M.G., Osband, I., Rabinowitz, N.C., György, A., Chiappa, S., Osindero, S., Teh, Y.W., Hasselt, H.V., Freitas, N., Botvinick, M.M., Legg, S.: Meta-learning of sequential strategies. ArXiv **abs/1905.03030** (2019)

[10] Grau-Moya, J., Genewein, T., Hutter, M., Orseau, L., Delétang, G., Catt, E., Ruoss, A., Wenliang, L.K., Mattern, C., Aitchison, M., et al.: Learning universal predictors. arXiv preprint arXiv:2401.14953 (2024)