Efficient Image Compression

James Camacho
MIT / AI
jamesc03@mit.edu

Linda He
Harvard / Applied Mathematics
lindahe@college.harvard.edu

Abstract

With high-dimensional spaces such as images or video, perfect communication becomes prohibitively expensive. A lossy, compressed version is cheaper to transmit and often good enough for most purposes. Traditional algorithms such as JPEG use a Fourier transform to pick out the most important features for transmission. In this paper, we explore using auto- and raster-encoders to automatically and efficiently compress images instead. We find they significantly outperform JPEG on the MNIST dataset, and discuss potential future improvements to their speed and cost via reinforcement learning.

1 Introduction

Over 75% of internet traffic comes in the form of video (Cisco, 2018), and YouTube alone nets $30bn from their distribution (Alphabet Inc., 2024). At these massive scales, every extra bit of compression is important. In this paper, we explore more optimal compression with the use of deep learning methods, including one adapted from language models.

Image and text have often been treated as separate domains, but there is much that can be applied between the two. Inspired by quantization and fractalization from image compression, Witten *et al.* proposed several lossy text encoding schemes as far back as 1992 (Witten et al., 2000). Recent advances in image generation such as denoising models (Ho et al., 2020) have similarly found applications in text generation by interpreting language tokens as pixels (Kou et al., 2024). Going the other direction, the popular Transformer architecture of language models have been applied to vision (Dosovitskiy et al., 2021), and has seen state-of-the-art success in video production (Liu et al., 2024).

The Vision Transformer algorithm bears a striking resemblance to JPEG, and this is no coincidence. Generation is the inverse of compression, and more generally "being able to compress well is closely related to acting intelligently" (Hutter, 2020). Unfortunately, this enforces a tradeoff between the algorithm's speed and size. For example, an unintelligent compressor may store images verbatim, or a generator may simply sample from its training dataset. Smarter algorithms will do much better, but require more computation.

This paper will focus on the tradeoff between image quality and compression length, but we will end with several suggestions for improving the running time via reinforcement learning.

2 Related Work

3 Data

MNIST, separate train and test sets.

4 Methods

4.1 JPEG

These vision transformers bear a striking resemblance to the JPEG algorithm

4.2 Auto-Encoders

Auto-encoders consist of an encoder and a decoder network trained together. Our encoder consists of three convolutional layers and a linear layer to control the latent dimension h, and the decoder has the reverse process (see Figure 1). By default, PyTorch stores floats using four bytes, so the encoding takes $4h$ bytes.

To train, we use the mean-squared error between the original and decoded images, running through two epochs of the MNIST training set with the Adam optimizer at $\gamma = 10^{-3}$. We keep a separate dataset for testing. Sample reconstructions can be seen in Figure 2. As expected, the quality improves as the latent dimension increases, though the reconstructions remain blurry. This is an artifact of our

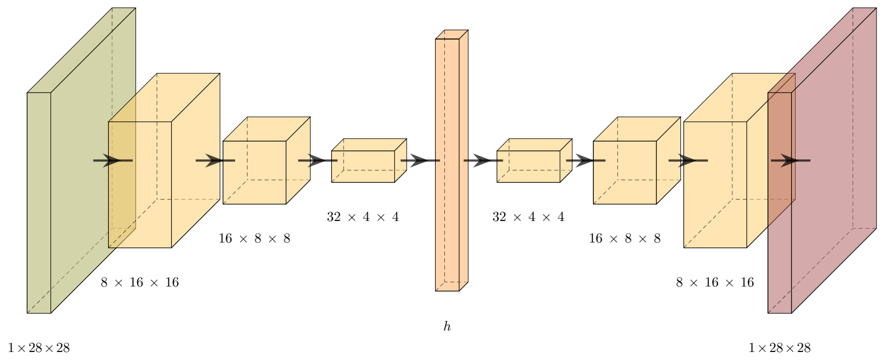

Figure 1: Auto-encoder network.

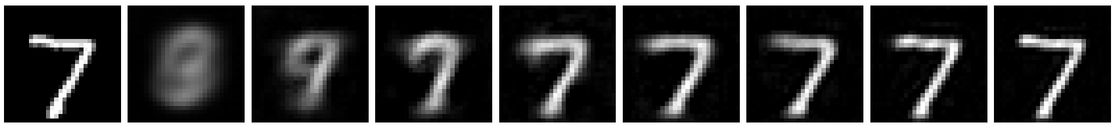

Figure 2: Reconstructions of a sample image (left). To the right, $h = 0, 1, 2, 4, 8, 16, 32, 64$.

loss function. Minimizing the squared error,

$$L_2 = \sum (x - \text{correct})^2$$

is equivalent to maximizing the log-likelihood under a Gaussian prior for the error function:

$$L_2 = \log \prod e^{(x-\text{correct})^2}.$$

The actual distribution of error is decidedly not Gaussian, so this biases the reconstruction towards a higher entropy than necessary. A common fix is to add a small adversarial loss (Makhzani et al., 2016), but since our quality metric is the L_2 loss, we opt to keep the bias. We will use this bias to improve our raster-encoder.

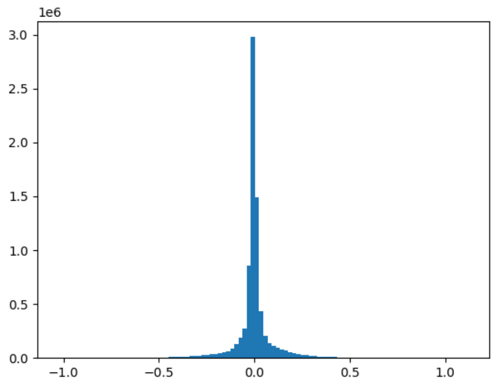

Figure 3: Error hisogram, 100 bins.

4.3 Raster-Encoders

In þe olde days, television was broadcast pixel by pixel, row by row, until an entire frame was drawn. This "raster" pattern flatten images into one-dimension, which we adopt to train in a language-model style.

Originally we converted the grayscale images to RGB and pulled apart the individual bits, so a $28 \times 28 \times 8$ bit grayscale image became a sequence of $T = 18816$ tokens. This is too large for attention, being over a gigabyte per head dimension ($4T^2 \approx 1.4$GB), so we used the state-space model Mamba (Gu and Dao, 2023). It uses a scan algorithm, so it uses $O(T)$ memory and runs in $O(T \log T)$ time, as opposed to $O(T^2)$ memory and time for attention.

Unfortunately, we ran into software/hardware issues when scaling up, and didn't want to implement Mamba from scratch, so we switched to Transformers. To reduce the sequence length, we kept our images grayscale and used bytes rather than bits for our tokens, giving a sequence of 784 tokens with 256 one-hot dimensions (Figure 4).

To compress, we can use arithmetic encoding on the output probabilities. Let $p(x)$ be the probability the next-byte prediction is x and $q(x)$ the distribution of bytes encountered (i.e. in the training set). We want to minimize the compression length,

$$\mathbb{E}_q[-\log p] = \sum q \log p,$$

i.e. the traditional cross-entropy loss. Since the MNIST dataset is relatively small, the training data is unlikely to perfectly match the underlying image distribution. To fix this, we can inject extra entropy into q while minimizing the expected error. We

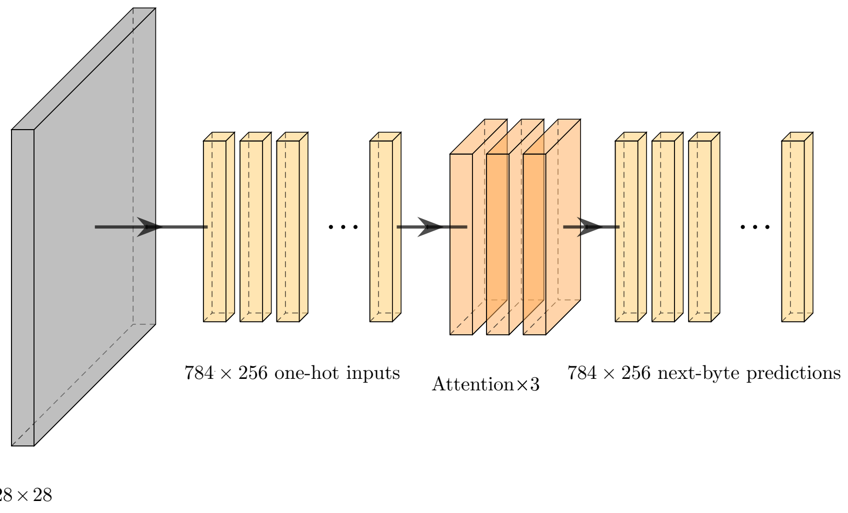

Figure 4: Raster-encoder network.

wish to minimize

$$L_2 = \mathbb{E}_q[(x - \text{correct})^2] = \sum (x - \text{correct})^2 q(x)$$

for a fixed entropy

$$H(q) = -\sum q \log q.$$

Lagrange multipliers give

$$q \propto e^{-\beta(x - \text{correct})^2},$$

which is the Gaussian blurring we saw from the auto-encoder section. The choice of inverse-temperature β does matter. Low values make q essentially uniform, while higher ones provide almost no smoothing. The former makes every pixel take nearly the full eight bits to compress, while the latter has too much surprisal for a few pixels. We find the best results for $\beta \approx 32$.

Since a decoder would not have access to the original image, it can only compute probabilities using previously decoded pixels. Thus, we have to encode the pixels raster-wise, feeding back in the chosen value after each step. We choose values to minimize the following cost function:

$$\text{cost}(x) = \text{quality} \cdot (x - \text{correct})^2 - \log p(x).$$

A higher quality ensures the error is smaller, at the cost of using more bits. Since the decoder does not have access to future pixel values, we also mask the encoder's attention to only look at previous pixels. This has the added benefit of removing artifacts present with full attention.

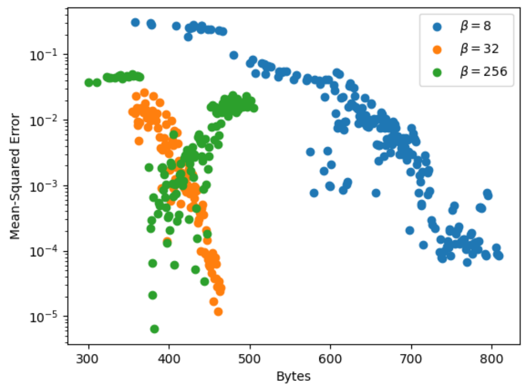

Figure 5: Raster-encoding for several smoothing parameters. Increasing β leads to better compression until reversing around $\beta = 64$, when the surprisal becomes too expensive to allow switching colors often enough.

5 Results

6 Discussion

Testing this: This is some text with a citation (Lazaridou et al., 2020).

Acknowledgments

References

Alphabet Inc. 2024. Alphabet 2024 q1 earnings report.

Cisco. 2018. Global device growth and traffic profiles.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image

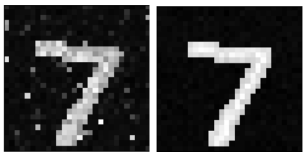
Figure 6: Full attention (left) vs. masked attention (right).

is worth 16x16 words: Transformers for image recognition at scale. *Preprint*, arXiv:2010.11929.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *Preprint*, arXiv:2312.00752.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Preprint*, arXiv:2006.11239.

Marcus Hutter. 2020. Universal artificial intelligence, aixi, and agi. Video interview.

Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. 2024. Cllms: Consistency large language models. *Preprint*, arXiv:2403.00835.

Angeliki Lazaridou, Anna Potapenko, and Olivier Tieleman. 2020. Multi-agent communication meets natural language: Synergies between functional and structural language learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7663–7674, Online. Association for Computational Linguistics.

Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, Lifang He, and Lichao Sun. 2024. Sora: A review on background, technology, limitations, and opportunities of large vision models. *Preprint*, arXiv:2402.17177.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2016. Adversarial autoencoders. *Preprint*, arXiv:1511.05644.

Ian Witten, Timothy Bell, Alistair Moffat, Craig Nevill-Manning, Cowlemon Tony, and Harold Thimbleby. 2000. Semantic and generative models for lossy text compression. *The Computer Journal*, 37.

A Example Appendix

This is an appendix.

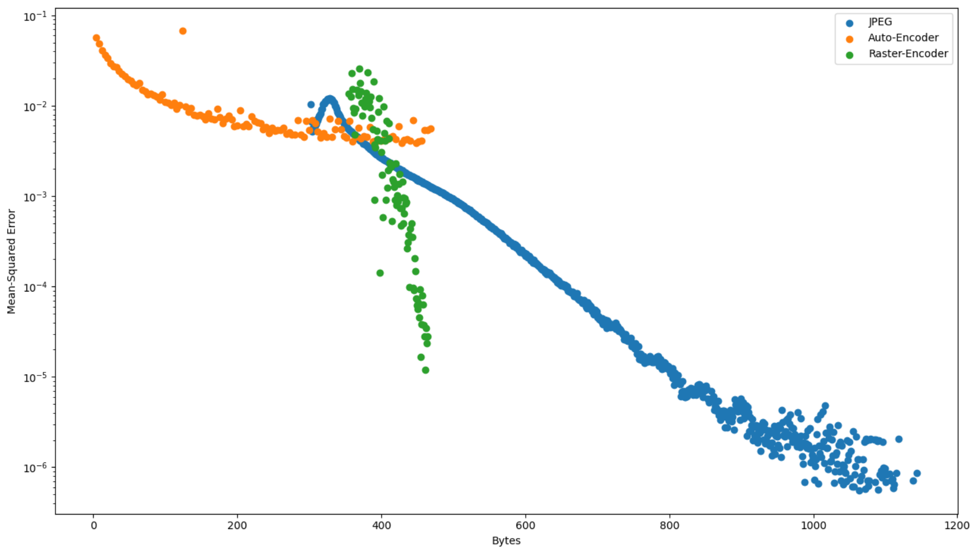

Figure 7: Compression results.