LEARNED TRANSFORM COMPRESSION WITH OPTIMIZED ENTROPY ENCODING.

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider the problem of learned transform compression where we learn both, the transform as well as the probability distribution over the discrete codes.

1 INTRODUCTION

We consider the problem of compressing data $\boldsymbol{x} \in \mathbb{X}$ sampled i.i.d. according to some unknown probability measure (distribution) $\boldsymbol{x} \sim \mu_x$[1]. We take the standard transform coding (Sayood, 2012) approach to compression where we first transform the data by a learned non-linear function, an encoder $\mathcal{E}_\theta : \mathbb{X} \to \mathbb{Z}$, into some latent representation $\boldsymbol{z} \in \mathbb{Z}$. We then quantize the transformed data $\boldsymbol{z} = \mathcal{E}_\theta(\boldsymbol{x})$ using a quantization function $\mathcal{Q}_{\boldsymbol{E}} : \mathbb{Z} \to \mathbb{C}$ parametrized by a learned embedding $\boldsymbol{E}$ (codebook/ dictionary) so that the discrete codes composed of indexes of the embedding vectors $\boldsymbol{c} = \mathcal{Q}_{\boldsymbol{E}}(\boldsymbol{z})$ can be compressed by a lossless entropy encoding and transmitted. The received and losslessly decoded integer codes are then used to index the embedding vectors and dequantize back to the latent space $\overline{\mathcal{Q}_{\boldsymbol{E}}} : \mathbb{C} \to \widehat{\mathbb{Z}} \subset \mathbb{Z}$ introducing a distortion due to mapping the codes only to the discrete subset $\widehat{\mathbb{Z}} \subset \mathbb{Z}$ corresponding to the quantization embedding. The dequantized data $\hat{\boldsymbol{z}} = \overline{\mathcal{Q}_{\boldsymbol{E}}}(\boldsymbol{c})$ are then decoded by a learned non-linear decoder $\mathcal{D}_\phi : \widehat{\mathbb{Z}} \to \mathbb{X}$ to obtain the reconstructions $\hat{\boldsymbol{x}} = \mathcal{D}_\phi(\hat{\boldsymbol{z}})$.

Our aim is to learn the transform (encoder/decoder) as well as the quantization so as to minimize the expected distortion $\mathbb{E}_{\mu_x} d(\mathtt{x}, \hat{\mathtt{x}})$[2] between the original and reconstructed data for some suitable distortion function $d : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$. However, we also wish to minimize the expected number of bits transmitted between the source and the receiver (the rate) when passing on the discrete codes $\mathbb{E}_{\mu_c} l(\mathtt{c})$, where l is the length of the bit-encoding. To control the trade-off between the two competing objectives we use a hyper-parameter λ

$$\mathcal{L} := \underbrace{\mathbb{E}_{\mu_x} d(\mathtt{x}, \hat{\mathtt{x}})}_{distortion} + \lambda \underbrace{\mathbb{E}_{\mu_c} l(\mathtt{c})}_{rate} \quad . \tag{1}$$

From standard results, e.g. Cover & Thomas (2006), the optimal length of encoding a symbol $\boldsymbol{c} \sim \mu_c$ is determined by Shannon's self information[3] $i_c(\boldsymbol{c}) = -\log p_c(\boldsymbol{c})$, where p_c is the discrete probability mass[4] of the distribution μ_c. Consequently, the expected optimal description length for the discrete code $\mathtt{c}$ can be bounded by its entropy $\mathbb{H}_{\mu_c}(\mathtt{c}) = -\mathbb{E}_{\mu_c} \log p_c(\mathtt{c})$ as $\mathbb{H}_{\mu_c}(\mathtt{c}) \leq \mathbb{E}_{\mu_c} l(\mathtt{c})^* < \mathbb{H}_{\mu_c}(\mathtt{c}) + 1$[5]. To minimize the rate we therefore minimize the entropy of the discrete code $\mathbb{H}_{\mu_c}(\mathtt{c})$.

$$\mathcal{L} := \mathbb{E}_{\mu_x} d(\mathtt{x}, \hat{\mathtt{x}}) + \lambda \mathbb{H}_{\mu_c}(\mathtt{c}) \quad . \tag{2}$$

[1]For the sake of generality, we do not take any assumptions about the probability space being discrete or continuous.

[2]We use $\mathtt{x}, \mathtt{z}, \mathtt{c}$ for the random variables and $\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{c}$ for their realizations.

[3]When considering bit-encoding, the log should be with base 2 instead of the natural base for nats.

[4]The probability mass p_c is the probability density function of μ_c with respect to the counting measure $\mu_c(\mathtt{c} \in \boldsymbol{A}) = \int_A p_c \, \mathrm{d}\# = \sum_{\boldsymbol{a} \in \boldsymbol{A}} p_c(\boldsymbol{a})$, such that $p_c(\boldsymbol{a}) = p_c(\mathtt{c} = \boldsymbol{a}) = \mu_c(\mathtt{c} = \boldsymbol{a})$.

[5]The '+1' in the upper bound is the result of naive rounding up of the self-information and can be reduced by more *clever* lossless compression strategy. This is, however, out of the scope of our paper in which we assume an off-the-shelf arithmetic coding-like algorithm.

2 QUANTIZATION

We employ the soft relaxation approach to quantization proposed in Agustsson et al. (2017) simplified similarly to Mentzer et al. (2018). However, instead of the scalar version Mentzer et al. (2018); Habibian et al. (2019) we use the **vector formulation of the quantization** as in van den Oord et al. (2017) in which the k quantization centers are the learned embedding vectors $\{\boldsymbol{e}^{(j)}\}_{j=1}^{k},\ \boldsymbol{e}_i \in \mathbb{R}^m$, columns of the $m \times k$ embedding matrix $\boldsymbol{E} = [\boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(k)}]$.

The quantizer $\mathcal{Q}_{\boldsymbol{E}}$ first reshapes the transformed data[6] $\boldsymbol{z} \in \mathbb{Z} \subseteq \mathbb{R}^d$ into a $m \times d/m$ matrix $\boldsymbol{Z} = [\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(d/m)}]$, then finds for each column $\boldsymbol{z}^{(i)} \in \mathbb{R}^m$ its nearest embedding and replaces it by the embedding vector index to output the d/m dimensional vector of discrete codes $\boldsymbol{c} = \mathcal{Q}_{\boldsymbol{E}}(\boldsymbol{z})$

$$\mathcal{Q}_{\boldsymbol{E}}: \quad \hat{\boldsymbol{z}}^{(i)} = \arg\min_{\boldsymbol{e}^{(j)}} \|\boldsymbol{z}^{(i)} - \boldsymbol{e}^{(j)}\|, \qquad c^{(i)} = \{j : \hat{\boldsymbol{z}}^{(i)} = \boldsymbol{e}^{(j)}\}, \quad i = 1, \ldots, d/m \ . \tag{3}$$

After transmission, the receiver recovers the quantized latent representation $\hat{\boldsymbol{Z}} = [\hat{\boldsymbol{z}}^{(1)}, \ldots, \hat{\boldsymbol{z}}^{(d/m)}]$ from $\boldsymbol{c}$ by indexing to the shared codebook $\boldsymbol{E}$ and decodes the data $\hat{\boldsymbol{x}} = \mathcal{D}_\phi(\hat{\boldsymbol{z}})$, $\hat{\boldsymbol{z}} = \mathrm{flatten}(\hat{\boldsymbol{Z}})$. As the discrete codes $\boldsymbol{c}$ are encoded losslessly, the recovered $\hat{\boldsymbol{z}}$ is exactly the one produced in the quantization step in equation 3. Therefore at training $\hat{\boldsymbol{z}}$ can be used directly by the decoder in the **forward pass** without triggering the indexing operation and passing through $\boldsymbol{c}$.

The quantization operation equation 3 of the transform $\boldsymbol{z} = \mathcal{E}_\theta(\boldsymbol{x})$ to the finite set of the embedding vectors is non-differentiable. To allow for the flow of gradients, we use a differentiable soft relaxation for the **backward pass** of the gradients

$$\tilde{\boldsymbol{z}}^{(i)} = \sum_{j}^{k} \boldsymbol{e}^{(j)} \mathrm{softmax}(-\sigma \|\boldsymbol{z}^{(i)} - \boldsymbol{e}^{(j)}\|) = \sum_{j}^{k} \boldsymbol{e}^{(j)} \frac{\exp(-\sigma \|\boldsymbol{z}^{(i)} - \boldsymbol{e}^{(j)}\|)}{\sum_{j}^{k} \exp(-\sigma \|\boldsymbol{z}^{(i)} - \boldsymbol{e}^{(j)}\|)} \ , \tag{4}$$

where instead of the hard encoding $\hat{\tilde{\boldsymbol{z}}}^{(i)}$ picking the single nearest embedding vector, the soft relaxation $\tilde{\boldsymbol{z}}^{(i)}$ is a linear combination of the embeddings weighted by their (softmaxed) distances with $\hat{\tilde{\boldsymbol{z}}}^{(i)} = \lim_{\sigma \to \infty} \tilde{\boldsymbol{z}}^{(i)}$ and the σ parameter can be used for an annealing strategy. The distortion loss is thus formulated as $d(\boldsymbol{x}, \hat{\boldsymbol{x}}) = d(\boldsymbol{x}, \mathcal{D}_\phi[\mathrm{sg}(\hat{\boldsymbol{z}} - \tilde{\boldsymbol{z}}) + \tilde{\boldsymbol{z}}])$, where sg is the stopgradient operator, which passes the gradients through the soft quantization in equation 4 to both the embeddings $\boldsymbol{E}$ and the $\mathcal{E}_\theta$.

The hard/soft strategy is different than the approach of van den Oord et al. (2017) where they use a form of straight-through gradient estimator by simply passing the gradients from the decoder input $\hat{\boldsymbol{z}}$ to the encoder output $\boldsymbol{z}$ and to train the embeddings they use a dedicated codebook and commitment terms using the ℓ_2 distance between the outputs of the encoder and the embedding vectors. This is also differenet from Williams et al. (2020), where they use the relaxed formulation of equation 4 for both forward and backward passes in a fully stochastic quantization scheme aimed at preventing the mode-dropping effect of a deterministic approximate posterior in a hierarchical vector-quantized VAE.

3 MINIMIZING THE CODE CROSS-ENTROPY

Though, as explained in section 1, the optimal lossless encoding is decided by the self-information of the code $-\log p_c(\boldsymbol{c})$, in our problem we cannot use this directly since the probability distribution μ_c (and therefore p_c) is unknown. To resolve this we can replace the unknown p_c by its estimated approximation q_c and derive the code length $\hat{l}(\boldsymbol{c})$ from $\hat{i}_c = -\log q_c(\boldsymbol{c})$. Instead of minimizing the expected self-information (the entropy $\mathbb{H}_{\mu_c}(\mathtt{c})$) in equation 2 we minimize the expected approximate self-information, the cross-entropy $\mathbb{H}_{\mu_c|q_c}(\mathtt{c}) = -\mathbb{E}_{\mu_c} \log q_c(\mathtt{c})$.

This approximation, however, yields inefficiencies as $\mathbb{H}_{\mu_c|q_c}(\mathtt{c}) \geq \mathbb{H}_{\mu_c}(\mathtt{c})$. More specifically, we can decompose the cross-entropy as

$$\mathbb{H}_{\mu_c|q_c}(\mathtt{c}) = D_{\mathrm{KL}}(p_c \| q_c) + \mathbb{H}_{\mu_c}(\mathtt{c}) \ , \tag{5}$$

[6]For notational simplicity we regard the data as d dimensional vectors. In practice, these are often $(d_1 \times d_2)$ matrices or even higher order tensors $(d_1 \times \ldots \times d_t)$. $\boldsymbol{z}$ can be seen simply as their flattened version with $d = \prod_i d_i$.

where D_{KL} is the Kullback-Leibler divergence between the two probability masses which can be interpreted as the expected additional bits over the optimal rate $\mathbb{H}_{\mu_c}(\boldsymbol{c})$ caused by using q_c instead of the true p_c.

In addition to the encoding $\mathcal{E}_\theta$, decoding $\mathcal{D}_\phi$ and quantization $\mathcal{Q}_{\boldsymbol{E}}$ functions described in section 1 we shall now therefore train also a probability estimator $\mathcal{P}_\psi : \{\boldsymbol{c}\}_i^n \to q_c$ by minimizing the cross-entropy $\mathbb{H}_{\mu_c|q_c}(\mathtt{c})$ so that the estimated q_c is as close as possible to the true p_c, the D_{KL} is small, and the above mentioned inefficiencies disappear.

As we cannot evaluate the expectation over the unknown distribution μ_c, we instead learn $\mathcal{P}_\psi$ by minimizing the empirical estimate of the cross entropy over the sample data equivalent to minimizing the negative log likelihood (NLL)

$$\arg\min_{\psi} -\frac{1}{n}\sum_{i}^{n} \log q_c(\mathtt{c} = \boldsymbol{c}_i), \quad \boldsymbol{c}_i \sim \mu_c \ . \tag{6}$$

Similar strategy has been used for example in Theis et al. (2017) and Ballé et al. (2017) both using some form of continuous relaxation of q_c as well as in Mentzer et al. (2018) using an autoregressive PixelCNN as $\mathcal{P}_\psi$ to model $q_c(\boldsymbol{c}) = \prod_i q_c(c_i | c_{i-1}, \ldots c_1)$.

There is one caveat to the above approach. In the minimization in equation 6 the sampling distribution μ_c is treated as fixed. Revisiting equation 5 we see that minimizing the cross-entropy in such a regime minimizes the D_{KL} and hence the additional bits of encoding incurred due to $q_c \neq p_c$. However, it does not anyhow impact the entropy $\mathbb{H}_{\mu_c}(\mathtt{c})$, which is treated as constant and therefore not optimized to minimize the rate as explained in section 1.

At first sight this may seem natural and inevitable since the samples $\boldsymbol{c}$ are the result of sampling the data $\boldsymbol{x}$ from the unknown yet fixed distribution μ_x. However, the distribution μ_c is not fixed. Rather it is determined by the learned transformation $\mathcal{T}_{\boldsymbol{E},\theta} = \mathcal{Q}_{\boldsymbol{E}} \circ \mathcal{E}_\theta, \ \mathtt{c} = \mathcal{T}_{\boldsymbol{E},\theta}(\mathtt{x})$ as the push-forward measure of μ_x

$$\mu_c[\mathtt{c} \in \boldsymbol{A}] = \mu_c[\mathcal{T}_{\boldsymbol{E},\theta}(\mathtt{x}) \in \boldsymbol{A}] = \mu_x[\mathtt{x} \in \mathcal{T}_{\boldsymbol{E},\theta}^{-1}(\boldsymbol{A})] \ , \tag{7}$$

where $\boldsymbol{A} \in \mathbb{C}$ and $\mathcal{T}^{-1}$ is the inverse image[7] defined as $\mathcal{T}^{-1}(\boldsymbol{A}) := \{\mathtt{x} \in \mathbb{X} : \mathcal{T}(\mathtt{x}) \in \boldsymbol{A}\}$. Changing the parameters of the encoder $\mathcal{E}_\theta$ and the embeddings $\boldsymbol{E}$ will change the measure μ_c and hence the entropy $\mathbb{H}_{\mu_c}(\mathtt{c})$ and the cross-entropy $\mathbb{H}_{\mu_c|q_c}(\mathtt{c})$ even with the approximation q_c fixed.

We therefore propose to optimise the encoder and the embeddings so as to minimize the cross-entropy not only through learning better approximation q_c but also through changing μ_c to achieve overall lower rate. Since the discrete sampling operation from μ_c is non-differentiable, we propose to use a simple continuous soft relaxation similar the one described in section 2. Instead of using the deterministic non-differentiable code assignments

$$p_c(\mathtt{c}^{(i)} = j) = \begin{cases} 1 & \text{if } \hat{\boldsymbol{z}}^{(i)} = \boldsymbol{e}^{(j)} \\ 0 & \text{otherwise} \end{cases}, \quad i = 1, \ldots, d/m \tag{8}$$

we use the differentiable soft relaxation

$$\hat{p}_c(\mathtt{c}^{(i)} = j) = \operatorname{softmax}(-\sigma \|\boldsymbol{z}^{(i)} - \boldsymbol{e}^{(j)}\|) = \frac{\exp(-\sigma \|\boldsymbol{z}^{(i)} - \boldsymbol{e}^{(j)}\|)}{\sum_j^k \exp(-\sigma \|\boldsymbol{z}^{(i)} - \boldsymbol{e}^{(j)}\|)} \ . \tag{9}$$

Our final objective is the minimization of the empirical loss composed of three terms: the distortion, the soft cross-entropy and the (hard) cross -entropy divergence

$$\widehat{\mathcal{L}}(\theta, \boldsymbol{E}, \phi, \psi) := \frac{1}{n}\sum_{i}^{n} d(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i) + \alpha\, s(\boldsymbol{c}_i) + \beta\, h(\boldsymbol{c}_i) \ . \tag{10}$$

The distortion may be the squared error $d(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2$ or other application-dependent metric (e.g. multi-scale structural similarity index for visual perception in images). Through the distortion loss we optimize the parameters of the decoder $\mathcal{D}_\phi$ and using the relaxation described in section 2 for the backward pass also the parameters of the encoder $\mathcal{E}_\theta$ and the quantization embeddings $\mathcal{Q}_{\boldsymbol{E}}$.

[7]The notation $\mathcal{T}^{-1}$ here should not be mistaken for an *inverse function* as $\mathcal{T}$ is generally not invertible.

The hard cross-entropy loss is

$$h(\boldsymbol{c}) = -\frac{m}{d}\sum_{j}^{d/m}\sum_{\boldsymbol{c}^{(j)}=1}^{k} p_c(\boldsymbol{c}^{(j)}) \log q_c(\boldsymbol{c}^{(j)}) = -\frac{m}{d}\sum_{j}^{d/m} \log q_c(\boldsymbol{c}^{(j)}) \quad , \tag{11}$$

where we treat the dimensions of the vector $\boldsymbol{c} = [\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(d/m)}]$ as independent so that the approximation q_c can be used directly as the entropy model for the lossless arithmetic coding (expects the elements of the messages to be sampled i.i.d. from a single distribution). The loss simplifies to the final form due to the 0/1 probabilities of the deterministic code-assignments in equation 8. Through the hard cross-entropy loss we learn the parameters of the probability model $\mathcal{P}_\psi$ outputting the q_c distribution.

The soft cross-entropy loss is

$$s(\boldsymbol{c}) = -\frac{m}{d}\sum_{j}^{d/m}\sum_{\boldsymbol{c}^{(j)}=1}^{k} \hat{p}_c(\boldsymbol{c}^{(j)}) \log \text{sg}[q_c(\boldsymbol{c}^{(j)})] \quad , \tag{12}$$

which uses the differentiable soft relaxation $\hat{p}_c$ of equation 9. This allows for back-propagating the gradients back to the encoder $\mathcal{E}_\theta$ and the quantizer $\mathcal{Q}_{\boldsymbol{E}}$. We use the stopgradient operator here to treat the q_c model as fixed in this part of the loss preventing further updating of the parameters of the probability model $\mathcal{P}_\psi$.

4 Experiments

As a proof of concept we conducted a set of experiments on the tiny 32x32 CIFAR-10 images (Krizhevsky, 2009). We use similar architecture of the encoder $\mathcal{E}_\theta$ and decoder $\mathcal{D}_\phi$ as Mentzer et al. (2018) (without the spatial importance mapping), with the downsumpling and upsampling stride-2 convolutions with kernels of size 4 and 10 residual blocks with skip connections between every 3. We fix the annealing parameter $\sigma = 1$ and the loss hyper-parameter $\beta = 1$[8]. We use ADAM with default pytorch parameters, one cycle cosine learning rate and train for 15 epochs. The code is available at: `https://bitbucket.org/dmmlgeneva/softvqae/`

We first compare the vector quantization (VQ) approach where the codebook is composed of m-long vectors versus the scalar (SQ) approach where it contains scalars $m = 1$ as e.g. in Mentzer et al. (2018). By construction, the VQ version needs to transmit shorter messages for the same level of downsampling. For example, with 8-fold downsampling to 4×4 latents $\boldsymbol{z}$ with 8 channels the discrete codes $\boldsymbol{c}$ of SQ have $d/m = 128$ elements. In VQ, the channel dimension forms the rows of the matrix $\boldsymbol{Z}$ with $m = 8$ and the $\boldsymbol{c}$ messages to be encoded and transmitted have only $d/m = 16$ elements. On the other hand, in the scalar version each of the 8 channels is represented by its own code and therefore allows for more flexibility compared to a single code for the whole vector.

Our preliminary experiments confirm the superiority of the VQ. In figure 1 left we plot the rate-distortion for comparable parts of the trade-off space. The VQ models use 2-folds downsampling resulting in $d/m = 16 \times 16 = 256$ long messages $\boldsymbol{c}$. The two curves are for embeddings (latent $\boldsymbol{z}$ channels) with size $m = 8$ and $m = 16$ respectively and the points around the curves are the result of increasing the size of the dictionary $\boldsymbol{E}$ as $k = [8, 16, 32, 64, 128]$ from left to right. The SQ models use 8-folds downsampling with latent $\boldsymbol{z}$ channels 8 and 16 resulting in $d/m = 4 \times 4 \times 8 = 128$ and $d/m = 4 \times 4 \times 8 = 256$ ($m = 1$ here). We observe that VQ can clearly achieves better trade-offs being in the bottom-left of the plots.

We next confirm the effectiveness of the soft cross-entropy term $s(\boldsymbol{c})$ in our final loss formulation in equation 10. Increasing the α hyper-parameter should put more importance on the rate minimization (through the entropy) as compared to the distortion. In the right plot of figure 1 we compare the points $k = [8, 32, 128]$ of the *'vq, 8'* curve for values $\alpha = [0.01, 0.001, 0]$ from left to right. With the highest $\alpha = 0.01$, the objective trade-off searches for the lowest rate tolerating higher distortion. The lower the α, the less we push for low rates which allows for smaller distortion. This behaviour

[8] In our preliminary experiments the results were not very sensitive to β. In fact, β influences only the speed with which the probability model $\mathcal{P}_\psi$ is trained compared to the other components of the model updated through the other parts of the loss.

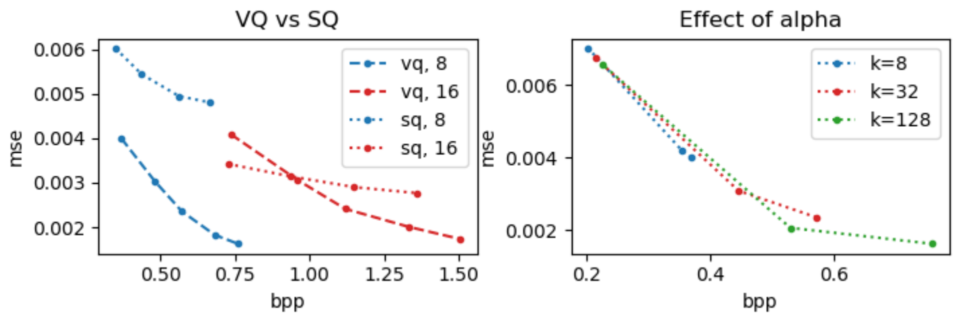

Figure 1: Rate-distortion curves. Rate is expressed in bits per pixel (bpp) of the original images, distortion is expressed in the mean squared error (mse) between the original and reconstructed images. See text for description.

corresponds well to the expected and desirable one where, as formulated in equation 1, we can now directly control the trade-off between between the two competing objectives by setting the hyper-parameter α.

Examples comparing the original with the reconstructed images for the *'vq, 8'* are available in the repo in the paper/pics folder. In the same place, there are examples of the learned q_c histograms for *'vq, 8, k=32'* with different values of α.

References

Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool. Soft-to-Hard Vector Quantization for End-to-End Learning Compressible Representations. *arXiv:1704.00648 [cs]*, June 2017.

Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end Optimized Image Compression. In *International Conference on Learning Representations*, 2017.

Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. Wiley, 2006.

Amirhossein Habibian, Ties van Rozendaal, Jakub M. Tomczak, and Taco S. Cohen. Video Compression With Rate-Distortion Autoencoders. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7032–7041, October 2019. doi: 10.1109/ICCV.2019.00713.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, CS, 2009.

Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional Probability Models for Deep Image Compression. In *CVPR*, 2018.

Khalid Sayood. *Introduction to Data Compression*. Elsevier, fourth edition, 2012. ISBN 978-0-12-415796-5. doi: 10.1016/B978-0-12-415796-5.00023-5.

Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy Image Compression with Compressive Autoencoders. In *International Conference on Learning Representations*, 2017.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural Discrete Representation Learning. In *Conference on Neural Information Processing Systems (NIPS)*, 2017.

Will Williams, Sam Ringer, Tom Ash, John Hughes, David MacLeod, and Jamie Dougherty. Hierarchical Quantized Autoencoders. In *Advances in Neural Information Processing Systems*, 2020.