Data Compression using Autoencoders and an Optimal Scalar Quantizer

Ahmed Gashgash [1] **Joseph Kim** [2]

Abstract

In lossy compression, the rate distortion function gives a lower bound on the minimum rate needed to reconstruct a source subject to a given distribution. For sources with a Gaussian distribution, this lower bound is analytically known. We train and test an autoencoder with Gaussian inputs and aim to compress and reconstruct the source with distortion close to that defined by the rate distortion function. We observe that autoencoders are far from optimal stand-alone compressors, but perform well when coupled with quantizers. Traditional image compression relies on handcrafted codecs that lack adaptability. Recently, researchers have been using deep neural networks (DNNs) to design compression systems. However, a common theme in these works is the use of primitive quantization schemes, such as rounding. In this project we propose using the Lloyd-Max algorithm for quantization, which is proven to minimize the expected distortion. We show that utilizing the Lloyd-Max quantization leads to better reconstruction performance across all bit-rates compared to simply rounding.

1. Introduction

One of the important outstanding problems in information theory is the development of practical codes for lossy compression of Gaussian sources at rates approaching Shannons rate-distortion bound (Venkataramanan et al., 2012). This lower bound specifies the minimum rate (bits per symbol) needed to reconstruct the source subject to a given distortion. This is an important task for compression engineers since a lot of source data can be modeled as being Gaussian distributed. Note that the Rate-Distortion function is mathematically derived for sources with a known probability distribution. It is used as a lower bound to evaluate the performance of compressors on these sources. For images, videos, and audio, there is no rate distortion function since these media types do not follow a closed form probability distribution. In the first part of this project, we explore how well an autoencoder neural network compresses Gaussian sources and whether it performs near optimally. Optimality in this case is achieving the rate distorion lower bound.

Traditionally, images are processed by a selected transformation and quantizer, followed by a lossless compressor. The quantizer essentially replaces numbers with representative values in a way that reduces size while still preserving most of the information contained in the image. Today, lossy image compression schemes use a hand crafted codec. The problem with this approach is that the codecs are manually generated and, therefore, would not adapt to new and emerging media types, such as Augmented Reality (AR) or Virtual Reality (VR); it would take years of research to develop well-performing codecs. However, in recent years, utilizing deep neural networks (DNNs) in compression systems has been an active area of research. For the rest of this paper, we will refer to this as deep compression.

In (Zhou et al., 2018) and (Theis et al., 2017) for example, an autoencoder is used to extract a lower dimensional latent vector, which is then quantized and losslessly encoded into a bit stream. Their approaches have achieved promising results. However, they are still trailing behind traditional codecs, especially in the low bitrate regime. A common theme in these studies is the use of primitive quantization methods for the latent vector, such as rounding. From a larger perspective, the loss incurred in lossy compression occurs at the quantization stage. Thus, selecting a near optimal quantizer is important to achieve better compression for a given bitrate. In this project, we propose to explore deep compression coupled with a more sophisticated quantizer.

In our compression system, we will have a neural network for both the encoder and decoder portions. Essentially, the encoder takes an image as an input and outputs a latent vector z. This vector z will be fed into a quantizer which outputs $\hat{z}$. A lossless arithmetic encoder will then generate a bit stream, which is our compressed file. Afterward, this bit stream is losslessly decoded and the output vector is fed into another neural network, which we call the decoder. The decoder then outputs the reconstructed image. The main contribution of our work is using an optimal scalar quantizer

*Equal contribution [1]Department of Electrical and Computer Engineering, Cornell University, Ithaca, New York [2]Department of Computer Science, Cornell University, Ithaca, New York.

(LLoyd-Max) and showing that it outperforms the primitive rounding uniform quantizer used by others.

2. Compressing Gaussian Sources

Many sources of data can be modeled with a Gaussian distribution, such as the coefficients of the Discrete Cosine Transform (DCT). In this section we consider an iid source sequence $\{X_i\}_i^n$ sampled from a normal Gaussian distribution:

$$f(x|0,1) = \frac{1}{\sqrt{2\pi}} e^{\frac{x^2}{2}}$$

We use an autoencoder to compress our source subject to mean squared error (MSE) distortion measure. In order to evaluate the optimality of our compression, we refer to the Rate-Distortion function of a Gaussian source.

2.1. Rate Distortion Theory

Given a source distribution and a distortion measure, the Rate distortion Theory problem is deriving the minimum rate description required to achieve a particular distortion, or equivalently, what is the minimum expected distortion achievable at a given rate. (Cover & Thomas, 2006)

Theorem: For a Gaussian Source $N(0,1)$ and squared error distortion, the Rate Distortion function is:

$$R(D) = \begin{cases} \frac{1}{2}\log\frac{1}{D} & \text{if} \quad 0 \leq D \leq 1 \\ 0 & \text{if} \quad D > 1 \end{cases}$$

2.2. Compression using Autoencoders

In our setup, we used an autoencoder with input dimension $n = 100$ and different latent dimensions d =10, 30, and 50. We trained the autoencoders on a dataset of Gaussian vectors that were randomly generated. The training set size was 20000 vectors and the test set was 5000 vectors that were generated in the same manner. Since the desired output was the reconstruction of the input, the loss function was the mean squared error of the following form:

$$L(X, \hat{X}) = \frac{1}{k}\sum_{i=1}^{k}(X_i - \hat{X}_i)^2$$

During training, the Adaptive Moment Estimation (Adam) method was used for back-propagation. Since the desired output $\hat{X} \in \mathbb{R}^n$ the last activation layer in the network was linear, whilst all other activation's were as follows:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

The Keras (Chollet et al., 2015) framework was used for this experiment.

2.2.1. Results

The models described above was trained for 500 epochs. Figure 1 summarizes the training and test loss for the different models used. It was observed that the loss converges to a value that is related to the latent dimension size. For example, when the latent dimension d = 50, the compression ratio is 2x, and the MSE loss incurred is 0.5. This is also observed for the other models with different latent dimension. We therefore believe that autoencoders fail to perform any stand alone meaningful compression, and perform no better than the naiive scheme. This naiive scheme is when the latent layer of size d, copies exactly d entries from the input and disregards the rest, and the compressor follows by reproducing exactly these d values and outputs 0's for the remaining values. Different adjustments were made to try and avoid this, such as adding random dropout between the layers, using different activation functions, and also trying a variational autoencoder (VAE) instead, however the results were exactly the same for all.

Traditionally, in lossy compression, the major loss component comes from quantization. It is difficult to allocate bits to compress real numbers between 1 and 2, for example, since this interval of real numbers is infinitely large. However, during quantization, we can split these intervals into levels, such as {1,1.25,1.5,1.75,2} and round every fraction to the nearest number in this set. This makes it easier to compress since we are only representing 4 values instead of infinitely many; two bits in this case suffice. However, since the whole process of quantization is inherently non differentiable, we believe this plays a major role in why the gradient descent trained autoencoders failed to produce any meaningful compression. These results are very insightful as to why it may be necessary to couple neural networks with quantizers in the application of compression, and not rely on neural network dimensionality reduction alone as a form of compression. The second section of this project addresses this.

3. Variational Autoencoder for Image Compression

Variational Autoencoders (VAEs) are a popular approach to unsupervised learning of complex distributions. They are appealing because they are built on top of standard function approximators (neural networks) and can be trained with stochastic gradient descent.

A VAE consists of an encoder and a decoder. The encoder $q_\theta(z|x)$ takes an input x and outputs parameters for the lower-dimensional latent representation z. The decoder $p_\phi(x|z)$ takes z and outputs the parameters for the input data distribution, giving a reconstruction of the input $\hat{x}$. The loss function of the VAE is the negative log-likelihood with

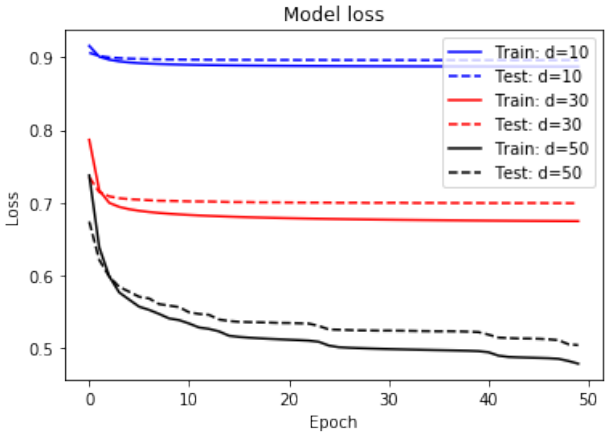

Figure 1. Model Loss

a regularizer. Because there are no global representations that are shared by all datapoints, we can decompose the loss function into only terms that depend on a single data point l_i. The loss function for each data point can be expressed as:

$$l_i(\theta, \phi) = -\mathbb{E}_{z\sim q_\theta(z|x_i)}[\log p_\phi(x_i|z)] + \mathbb{KL}(q_\theta(z|x_i)||p(z))$$

The first term is the reconstruction loss and the second term is the Kullback-Leibler divergence which acts as a regularizer and penalizes the model if the posterior does not match up with a particular prior distribution. If we set the prior distribution $p(z)$ to the standard normal $\mathcal{N}(0, 1)$, then there is a closed, analytical expression for the $\mathbb{KL}$-divergence (Kingma & Welling, 2014).

Traditionally, VAEs are used as a generative modeling procedure, though they have also recently been used for image compression (Balle et al., 2018; Zhou et al., 2018). We will adopt network architectures similar to those described in these studies.

The lower-dimensional feature space determined by the VAE contains information of the input and can be seen as a form of lossy compression. However, as we have seen previously, they do not work well by themselves. A key difference in our approach is to use a new framework where the latent representation from the encoding neural network undergoes state of the art lossy compression via a suitable quantizer. One such quantizer is the Lloyd-Max algorithm, which is proven to minimize the expected distortion. The limitation of this quantizer is that its rate-distortion performance is largely dependent on the approximation of the probability distribution over the latent input. We believe that using a VAE and an associated KL-divergence latent loss will effectively allow us to rely on the Lloyd-Max quantizer to achieve more optimal rate-distortion compared to previous approaches that also utilized VAEs but with primitive quantizers.

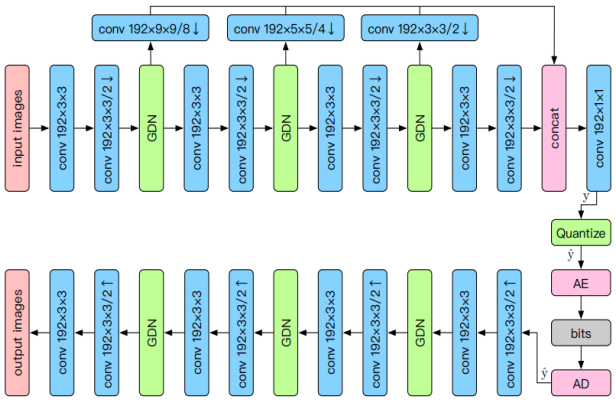

Figure 2. Illustration of the VAE architecture used in Zhou et al. (2018). Convolution parameters are denoted as number of filters × kernel height kernel width / down- (↓) or up- (↑) sampling stride. AE and AD represent lossless arithmetic encoder and decoder, respectively.

3.1. Lloyd-Max Quantizer

The Lloyd-Max algorithm gives the optimal quantization of a random variable. Given a random variable X and a set of reconstruction points $\{\hat{X}(w)\}$, the distortion is minimized if the random variable X is mapped to the closest representation $\hat{X}(w)$. These reconstruction points (levels), and the regions between them are defined by the quantizer that is used. It makes sense to allocate more levels where the random variable has a higher probability of occurring. For a uniform random variable, a uniform quantizer in which the spacing between the regions is uniform is optimal. However for other random variables, these regions and levels are not uniform. Lloyd Algorithm solves this problem and defines the optimal regions and levels given the number of reconstruction levels and the probability distribution of the random variable. It achieves this iteratively as follows: its starts with an initialized set of reconstruction points and finds the optimal set of reconstruction regions, these are the nearest neighbor regions with respect to the distortion measure used, it then finds the optimal reconstruction points or levels for these regions, in our case (MSE) the centroid of the region, and repeats until convergence. The algorithm will converge to a local minimum of the distortion (Cover & Thomas, 2006). The following equation are solved iteratively to specify the regions q and the level or reconstruction points x given the probability distribution $f_x(x)$ and the number of levels L:

$$q_k = \frac{\int_{x_k}^{x_{k+1}} x f_x(x) dx}{\int_{x_k}^{x_{k+1}} f_x(x) dx}$$

for $k = 1, .., L$

$$x_k = \frac{q_{k-1} + q_k}{2}$$

for $k = 2, .., L$

4. Experiments

A model was trained (50000 images) and evaluated (10000 images) using the CIFAR-10 dataset (Krizhevsky, 2009) of 32×32 RGB images. The model was trained for 200 epochs using a batch size of 64 and initialized with a learning rate of 0.0002, which was halved every 30 epochs. All models were optimized with ADAM. The latent representation is encoded as a 512-dimensional vector. All code was written in Python and the models were developed using PyTorch.

The models were evaluated three different ways. The first is the baseline behavior of using no lossy quantizer. The second utilized a Lloyd-Max quantizer for a specified number of bits $b = 3, 4, 5, 6$ and a standard normal distribution as the target distribution. The third used a rounded uniform quantizer that split the interval $[-3, 3]$ into 2^b equally spaced intervals and rounded the latent representation to the closest number within this set of intervals. The bounds -3 and 3 were chosen because the standard normal distribution contains almost all of its probability density within three standard deviations.

Note that in the loss expression that was clarified earlier, there is a reconstruction loss term as well as a regularization term. The reconstruction loss can take on a number of different forms such as mean-squared error (MSE), peak signal to noise ratio (PSNR), or structural similarity index (SSIM) (Wang et al., 2004). Previous studies in deep compression don't have a consensus on which reconstruction loss to optimize. The first task in this project was to determine which metric would be better for our task. We compared the results of the model when optimizing for MSE and SSIM (note that PSNR depends on MSE, and thus, is not included as it would be a redundant comparison):

$$\text{MSE}(x, \hat{x}) = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} [\hat{x}(n, m) - x(n, m)]^2$$

$$\text{SSIM}(x, \hat{x}) = \frac{(2\mu_x \mu_{\hat{x}} + c_1)(2\sigma_{x\hat{x}} + c_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + c_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + c_2)}$$

4.1. Results

Figure 3 highlights the loss curves when training the model and optimizing for MSE or SSIM. The images in figure 4

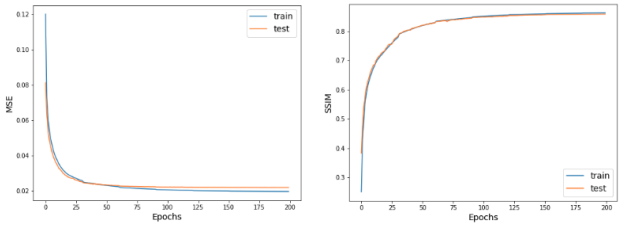

Figure 3. Loss curves comparing training while optimizing MSE (left) and SSIM (right). The blue curves are for the training set and the orange curves are for the test set.

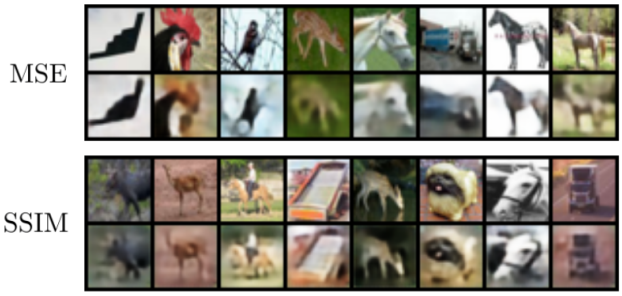

Figure 4. Original and reconstructed images generated when optimizing for MSE or SSIM. The reconstructed images utilize 4-bit Lloyd-Max quantization.

indicate that SSIM is better at constructing images that are deemed higher quality to the human eye. There is some valid intuition for why this may be true. Consider the following scenario. Take an image and shift it over by one pixel to the right. The image itself would look almost exactly the same, but the MSE would be relatively large. Since the mean and variance of both images is the same, the only term that affects the SSIM metric is the covariance between the original image and the image after the shift. However, since two adjacent pixels are highly correlated, the covariance term in the numerator will be relatively high, leading to a high SSIM (SSIM values are between 0 and 1, where 1 is perfect reconstruction).

All models are now optimized for SSIM. Table 1 compares the MSE and SSIM between the various quantization schemes described previously at various representation levels. The representation levels are the number of bins for quantization and is determined as 2^b where b is the number of bits allowed for each element in the low-dimensional latent representation. Clearly, using too low of a bit-rate would not achieve high performance. in addition, we see that there is a large gap at low bit-rates between using the Lloyd-Max quantizer and the rounding quantizer, while this gap tends to disappear at higher bit-rates. This makes sense because at low bit-rates, the rounding quantizer will spread the representations uniformly while the Lloyd-max quantizer will have representations concentrated in the dense

Model	Rep.	MSE	SSIM
Baseline	-	0.0216	**0.8586**
Lloyd-Max	8	0.0315	0.8217
	16	0.0289	0.8451
	32	0.0240	0.8513
	64	0.0231	**0.8571**
Uniform Rounding	8	0.0419	0.7933
	16	0.0312	0.8321
	32	0.0265	0.8461
	64	0.0232	**0.8548**

Table 1. This table compares the MSE and SSIM values for the various quantization methods after completing training. The model was trained to optimize SSIM.

region of the standard normal distribution. As the number of representation levels increases, the differences between quantized representations of numbers tend to decrease even when far from the center of the Gaussian.

For the case of using 4-bit quantization, we achieve $3 \times 32 \times 32 \times 8/512 \times 4 = 12$ times compression before even considering the lossless compression from the arithmetic encoder. The numerator is the product of the number of color channels (3), the size of the images (32×32), and the size of an unsigned 8-bit int (8). The denominator is the product of the size of the latent representation (512) and the number of bits allowed for quantization (4). Standard JPEG compression achieves somewhere between 4-10 times compression, depending on the JPEG parameters. Unfortunately, due to the lack of public code for customizable JPEG compression, we were unable to make a direct comparison to JPEG. However, Fig. 4 indicates that 4-bit quantization already performs extremely well.

5. Future Work

We noticed that neural networks perform very well in the transformation stage of lossy compression but are unable to perform quantization. The next step of this work would be to extend our autoencoder network to operate on higher resolution images. Unfortunately, due to hardware constraints, we were unable to train our model on datasets of larger images such as the Youtube8M dataset and the ImageNet dataset. This would be a big step forward since much of the distortion when using the CIFAR-10 dataset is from areas of the image with a high density of edges. Since 32×32 images are relatively small in size, edges are not captured without appearing grainy in the images. Training the model on higher resolution images would most likely allow the model to learn how to reconstruct theses edge regions better.

6. Conclusion

In this study, we first explored how well autoencoders can compress Gaussian sources when used as standalone compressors. We showed that autoencoders at best perform no better than the naiive scheme and achieve no compression. We then explored coupling autoencoders with an optimal quantizer and showed that it outperformed currently used rounding methods when compressing images. Though we could not compare to standard compressors such as JPEG due to lack of public code, we show that even at higher levels of compression without also utilizing a lossless encoder, we achieve relatively good reconstruction.

References

Balle, J., Minnen, D., Singh, S., Hwang, S. J., and Johnston, N. Variational image compression with a scale hypyerprior. *International Conference on Learning Representations*, 2018.

Chollet, F. et al. Keras. https://keras.io, 2015.

Cover, T. M. and Thomas, J. A. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, New York, NY, USA, 2006. ISBN 0471241954.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *arXiv e-prints*, 2014. URL https://arxiv.org/abs/1312.6114.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Theis, L., Shi, W., Cunningham, A., and Huszr, F. Lossy image compression with compressive autoencoders. 03 2017.

Venkataramanan, R., Joseph, A., and Tatikonda, S. Gaussian rate-distortion via sparse linear regression over compact dictionaries. In *2012 IEEE International Symposium on Information Theory Proceedings*, pp. 368–372, July 2012. doi: 10.1109/ISIT.2012.6284210.

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13, 2004.

Zhou, L., Cai, C., Gao, Y., Su, S., and Wu, J. Variational autoencoder for low bit-rate image compression. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.