Theory and Practice of Vector Quantizers Trained on Small Training Sets*

Tech. Report UW-CS&E TR-92-12-08

David Cohn
Dept. of Brain & Cog. Sci.
E10-243, MIT
Cambridge, MA 02139

Eve A. Riskin
Dept. of Electrical Eng.
FT-10, Univ. of Washington
Seattle, WA 98195

Richard Ladner
Dept. of Comp. Sci. & Eng.
FR-35, Univ. of Washington
Seattle, WA 98195

Abstract

We examine how the performance of a memoryless vector quantizer changes as a function of its training set size. Specifically, we study how well the training set distortion predicts test distortion when the training set is a randomly drawn subset of blocks from the test or training image(s). Using the Vapnik-Chervonenkis dimension, we derive formal bounds for the difference of test and training distortion of vector quantizer codebooks. We then describe extensive empirical simulations that test these bounds for a variety of bit rates and vector dimensions, and give practical suggestions for determining the training set size necessary to achieve good generalization from a codebook. We conclude that, by using training sets comprised of only a small fraction of the available data, one can produce results that are close to the results obtainable when all available data are used.

1 Introduction

Vector quantization (VQ) [7, 8] is a data compression technique that can be used to reduce the storage or transmission costs of binary and grayscale images. It is *lossy* in that the compressed/uncompressed image is a degraded copy of the original image. A major part of the computational cost in VQ is designing the VQ codebook used to encode the image. This design is usually done by "training" a codebook on a set of images that is somehow representative of the images to be encoded.

It is generally presumed that the more data that are used to design a VQ codebook, the better the codebook will encode its test images. The data consist of blocks of pixels extracted from a training image. We will alternatively refer to these as *training blocks* or *training vectors*. A training set of ten 512×512 pixel images, broken into 4×4 blocks, has 163,840 blocks available for training. Encoding a grayscale image at one half bit per pixel requires approximating all these 4×4 vectors with a codebook of 256 representative vectors. Statistically, the distribution of vectors in the image will be well represented even in a small random sub-sample of the total available training set. Since the computational cost of codebook design is linearly related to the size of the training set, we would like to be able to determine at what point the diminishing returns of a larger training set size are outweighed by the additional time required to train on it.

The purpose of this paper is to investigate the size of training sets that are sufficient for the construction of codebooks which are nearly as good as codebooks trained on all the available data. We describe both theoretical and empirical results. In theory, we show how the VQ problem can be viewed as a learning

*Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence as a REGULAR PAPER. Correspondence to D. Cohn. Electronic mail: cohn@psyche.mit.edu

problem. This allows us to derive upper bounds, from bounds already known in learning theory, on the size of the training set needed to build good codebooks based on the size of the codebook and the dimension of the vectors in the codebook. Unfortunately, these bounds are too general to give useful advice to practitioners. Fortunately, this learning theory analysis of the VQ problem suggested an empirical study which eventually led us to the results which form the basis of our practical advice.

As suggested by our learning theory analysis, the right thing to look at is how the value of *(test - train)* decreases as the size of the training set increases. The value *(test - train)*, which we call the *generalization curve*, is simply the difference between the test error of a codebook and the training error of the same codebook. If we train on larger and larger samples of an image (or set of images) the generalization curve approaches zero. Surprisingly, the generalization curve, as shown in our empirical studies, has a very simple functional, namely α/m, where α is a constant and m is the size of the training set. We call the constant α the *learning complexity* of the codebook for the image (or set of images) because it is the main determinant of how large a training set is needed to build a good codebook. For example, the value of α is less than 50 for codebooks of size 256 of 16 dimensional vectors for typical images. If one desires a 1% difference between testing and training error, then the size of the training set need not be larger than $50/.01 = 5,000$. This training set amounts to only 30% of an entire 512×512 pixel image. If the training is for a set of 10 images rather than a single image, then again only 5,000 vectors are needed in the training set which represents only 3% of potential training vectors. The problem of training set size has also been studied independently at Stanford University [4] with different but consistent results.

We summarize the remainder of the paper as follows. Section 2 of this paper provides a brief introduction to vector quantization and details the derivation of bounds on training set sizes for VQ codebooks. In Section 3, we empirically examine the generalization error (the difference between test and training distortion) of VQ codebooks with respect to this bound. We find that the formal "worst-case" bounds derivable from the theory are not tight enough to provide practical guidance for codebook design, and so describe empirically-derived "average case" and "worst case" performance. In both cases, the generalization error is found to approach zero inversely proportional to the size of the training set. In Section 4, we examine the case where the training and testing sets differ, a common practical occurrence. We find that, although the theory in Section 2 is unable to make any predictions about this case, the empirical results of Section 3 appear to apply. Finally, in Section 5, we discuss practical guidelines deriving from this work indicating that by using training sets comprised of only a fraction of the available data, one can produce results that are close to the results obtainable when all available data are used. We also suggest methods that researchers may use to determine appropriate training set sizes for their own VQ problems.

2 Vector quantization and the Vapnik-Chervonenkis dimension

When applying standard VQ to an image, the image is first broken up into (typically) rectangular pixel blocks (e.g. 4×4 pixels). Each of these blocks is a k-dimensional vector. The image is "quantized" by assigning to each of its blocks the "closest" vector (by some metric) in a small number of predetermined vectors. This reduced set of vectors is the *codebook* that is used to encode the image. We may then simply store or transmit the index of the selected codebook vector for every block rather than storing or transmitting the entire block. For example, encoding a grayscale image (eight bits per pixel) with a codebook of 256 4×4-pixel blocks will require that $\log_2(256) = 8$ bits be used for every 128-bit block, resulting in a 16 to 1 compression ratio. We will, for the remainder of this paper, indicate compression as the number of bits per pixel b, or *bit rate*.

This quantization imposes some degradation on image quality, the extent of which is governed by the distribution of blocks in the k-dimensional vector space, the size of the codebook, and the care with which these codebook vectors are chosen. A new codebook may be designed for each source image, or a single codebook may be designed to quantize a large number of images of some class. Given an image (or set of images), a fixed block size, and a fixed codebook size, iterative algorithms such as the Generalized Lloyd Algorithm (GLA) select codebook vectors which locally optimize some image degradation measure [12]. Typical distortion measures are the mean-squared error, the weighted mean-squared error, and the Itakura-

Saito distortion (for speech) [10].

Below, we first introduce the pattern classification problem and formal bounds that have been derived for it using the Vapnik-Chervonenkis dimension. We then show how these bounds may be used to bound the difference in training and test performance of a VQ codebook.

2.1 Pattern classification and the VC-dimension

The results of [1, 16, 17] concern the asymptotic performance of learning systems. More specifically, these results bound the difference between the empirically observed performance of a system and its "true" performance as a function of the number of inputs over which the empirical performance was observed. In this paper we will be concerned with these results as they apply to pattern classification.

The pattern classification problem is this: we are given a domain X, with an associated unknown probability density $\mathcal{P}$. There is an unknown subset $c' \subseteq X$ (called the *target concept*) which we wish to learn. The $I_{x \in c'}(x) = 1$ indicates that a point $x \in X$ is in c'; $I_{x \in c'}(x) = 0$ indicates that x is not in t.

Now let us consider hypothesis concept $c \subseteq X$. We define the generalization error, or *error rate*, of c with respect to a fixed target concept and distribution pair $(c', \mathcal{P})$ as

$$\epsilon(c, (c', \mathcal{P})) = Pr[c(x) \neq I_{x \in c'}(x)], \text{ for } x \text{ drawn according to } \mathcal{P} \tag{1}$$

where $Pr[A]$ denotes the probability of event A. Based on information from training examples, we attempt to choose a concept c that minimizes the error rate. Note that this is a two-sided error measure; all points in c that are not in c' are in error, as are all points in c' that are not in c. In most cases, the error minimization is done by making an empirical estimate of $\epsilon(c, (c', \mathcal{P}))$ and choosing the concept with the lowest empirical error. The simplest way to measure the empirical estimate is by measuring the empirical error over a sample $\mathcal{P}^m = (x_0, x_1, \ldots, x_m)$ of m points drawn from $\mathcal{P}$. We write this as

$$\epsilon(c, (c', \mathcal{P}^m)) = \frac{1}{m} \sum_{i=1}^{m} \begin{cases} 0 & c(x_i) = I_{x \in c'}(x_i) \\ 1 & \text{otherwise.} \end{cases} \tag{2}$$

Other more involved methods of estimating empirical error are discussed in [18].

Typically, our hypothesis is chosen according to some rule, such as "all points that are within Euclidean distance 1 of point z." This rule defines a *concept class*, C. The diversity of hypotheses in the class, and the class' representational power may be indexed by the *Vapnik-Chervonenkis dimension*, or VC-dimension of the class.

The VC-dimension of a concept class is defined as follows: Consider a set of m points $S^m = (x_1, x_2, \ldots, x_m)$, and a concept c. The concept classifies each point as either 1 or 0, and thus imposes a labeling on the set. For a set of m points, there are 2^m possible labelings. We will say that concept class C *shatters* set S^m if there exists a concept in C that imposes each of the 2^m possible labelings on S^m. The VC-dimension of the class is the size m of the largest set S^m that can be shattered by C. For example, the VC-dimension of the class of balls in k-dimensional Euclidean space is $3k$ [1].

Given the empirical error $\epsilon(c, (c', \mathcal{P}^m))$, for c selected from a class with VC-dimension d, the theorems of Vapnik & Chervonenkis bound the probability that the $\epsilon(c, (c', \mathcal{P}))$ will exceed some value. Specifically, if $m > d/2$, then with probability $\geq 1 - \delta$

$$\epsilon(c, (c', \mathcal{P})) \leq \epsilon(c, (c', \mathcal{P}^m)) + q \left(1 + \sqrt{1 + \frac{2\epsilon(c, (c', \mathcal{P}^m))}{q}} \right), \tag{3}$$

where

$$q = \frac{d}{2m}(\ln \frac{2m}{d} + 1) - \frac{\ln \delta}{2m} \text{ [16].}$$

Another popular way of expressing this bound is as the difference between $\epsilon(c, (c', \mathcal{P}))$ and $\epsilon(c, (c', \mathcal{P}^m))$, the testing and training errors, so that the last term in Equation 3 describes a bound on the learning curve.

For example, let us assume we have selected a concept c from a class with VC-dimension $d = 10$, based on $m = 1000$ training examples. If our error on the training examples was 3% ($\epsilon(c, (c', \mathcal{P}^m)) < 0.03$), then with 95% confidence ($\delta = 0.05$), the true error of c is less that 10.14%. A detailed description of the VC-dimension and its use in formal learning theory is beyond the scope of this paper, but may be found in [1, 16].

2.2 Framing VQ as a Classification Problem

The bound in the previous subsection is useful because it lets us predict generalized performance based on observed performance. Applied to VQ, it would allow predicting the distortion of a codebook on some image based on its performance on just a small part of the whole image. This would have great computational advantages when designing a codebook, since training time depends linearly on the training set size.

To use the above theorems for vector quantization, we must be able to frame VQ as a classification problem. The first step is to define X, the domain of the problem. If we are interested in encoding binary images using k-dimensional vectors, then the domain is simply $X = \{0, 1\}^k$, the space of all k-dimensional binary vectors. If our interest is in 8-bit grayscale images with k-dimensional vectors, then our domain would be $X = \{0, 1, \ldots, 255\}^k$. Each point in the domain represents a k-dimensional block that might appear in an image.

With respect to a memoryless vector quantizer, every image (or set of images) may be viewed as just an unordered set of points in this domain. This set corresponds to the formal notion of a "concept" in learning theory. The frequency with which these points occur in the images defines the distribution $\mathcal{P}$ over the domain. Note that each concept in our formal model corresponds to many possible images. One can imagine shuffling the ordering of the blocks in an image – for memoryless quantizers, all possible permutations of the blocks are equivalent.

Our concern in choosing the VQ codebook is that it encode an image with the least possible error. We will begin by describing a simple "tolerance" error measure that fits well in the concept learning framework; in the following subsection we will extend the model to include more practically useful error measures.

2.3 The "tolerance" measure

In a classification problem, one measures error in terms of the probability of an example being classified correctly or incorrectly. The most popular VQ distortion measures measure *how far* from correct a given encoding of a block is. To reconcile these two approaches, we define a simple VQ distortion measure which we shall refer to as the *tolerance* measure. Simply stated, a VQ codebook will be said to have zero error on some block if it encodes that block within some "tolerance" (which we describe below). The error of the codebook on an image is the probability that it will encode a random block from that image to within the specified tolerance.

2.3.1 Binary images

We begin our analysis by considering binary (black/white) images. Consider a point $x \in X$. We will say that a codebook (which we can think of as a *concept*) c *covers* x if there is a vector v in c such that the Hamming distance between x and v (written $\mathcal{H}(v, x)$), is at most r bits:

$$cover_r(c, x) = \begin{cases} 1 & \exists v \in c : \mathcal{H}(v, x) \leq r \\ 0 & \text{otherwise.} \end{cases}$$

The *r-tolerance* error of a codebook on an image is the probability that a block drawn at random from the image S (that is, according to $\mathcal{P}_S$) will fail to be encoded within r bits.

$$\epsilon_r(c, S) = Pr[cover_r(c, x) = 0], \text{ for } x \text{ drawn according to } \mathcal{P}_S. \tag{4}$$

An empirical estimate of $\epsilon_r(c, S)$ may be made based on a sample $S^m = (x_1, x_2, \ldots, x_m)$ of m blocks drawn at random from S:

$$\epsilon_r(c, S^m) = \frac{1}{m} \sum_{i=1}^{m} \begin{cases} 0 & cover_r(c, x_i) = 1 \\ 1 & \text{otherwise.} \end{cases} \tag{5}$$

The "concepts" we can represent are those sets of points that can be covered by a codebook of N k-dimensional vectors using the r-tolerance criterion. Technically, the target concept is the set of vectors that appear in the image S. We simplify the problem by defining the target concept to be the entire domain X. We can do this because there should be no penalty for covering a vector that does not occur in an image, only a penalty for *not* covering a vector that does occur. Since vectors that do not occur in the image have zero probability of being drawn in a random sample, the error measures of the two concepts will be identical.

The "learning" part of vector quantization involves finding a codebook with minimal or near-minimal error on an image. For the moment, we will concern ourselves with minimizing the r-tolerance error for some given r. Typically, one specifies the vector dimension and number of vectors in the codebook in advance, based on the desired bit rate and the encoding complexity that can be tolerated. This defines the hypothesis class out of which we will select our "hypothesis." We will use the following notation:

$C_{N,k}$ = class of all codebooks of N k-dimensional vectors
$c_{N,k}$ = a particular codebook.

We are given a *training set*, a set of vectors over which the training algorithm will attempt to minimize the empirical error. If the training set is sufficiently large and representative of the images that we wish to encode, then this "learned" codebook should provide near-minimal error on these test images. If one has a fixed set of images and sufficient computational power, it is straightforward to use the entire image as a training set. What we investigate in the remainder of this paper is how codebooks that have been trained on only a small fraction of the available data will perform on the bulk of the image(s). We denote the r-tolerance errors of a codebook as follows:

$\epsilon_r(c, S)$ = r-tolerance error rate of codebook c on image S
$\epsilon_r(c, S^m)$ = r-tolerance error rate of codebook c on m blocks drawn from image S.

2.3.2 Grayscale images

To fit grayscale images into the tolerance model, we need an additional parameter. In binary images, a given pixel is either correct or incorrect when compared to a codeword. In a grayscale image, each pixel has some real-valued distortion with respect to a codeword. To accommodate this, we define a parametric threshold t. Now, for a grayscale image, under the (r, t)-tolerance, a given block has zero error if no more than r of its pixels has distortion greater than t. This t can be defined arbitrarily for whichever pixel distortion measure is of interest to us. Once this new parameter is set, it is straightforward to extend the above analysis of binary images to grayscale images:

$\epsilon_{r,t}(c, S)$ = (r, t)-tolerance error rate of grayscale codebook c on image S
$\epsilon_{r,t}(c, S^m)$ = (r, t)-tolerance error rate of grayscale codebook c on m blocks drawn from image S

2.4 Determining the VC-dimension of a VQ codebook

When we speak of the VC-dimension of a codebook, we are actually referring to the VC-dimension of the *class* of codebooks meeting some specification. In this case, we mean the class of all k-dimensional N-vector codebooks using a specific tolerance error measure. We denote this as:

$d(N, k, r)$ = VC-dimension of binary codebook class $C_{N,k}$ using the r-tolerance error measure
$d(N, k, r, t)$ = VC-dimension of grayscale codebook class $C_{N,k}$ using the (r, t)-tolerance error measure.

We say that a class of such codebooks *shatters* a set S of vectors if there exists a codebook in that class that covers each of the $2^{|S|}$ possible subsets of S (including the empty set). Then the VC-dimension of a class of codebooks is just the cardinality of the largest set of vectors that can be shattered by the class. For brevity, in the remainder of this paper, we will simply refer to the VC-dimension of a codebook with the understanding that this dimension formally applies to the class of which the codebook is a member.

The simplest upper bound we can place on the VC-dimension of a binary VQ codebook class is derived by a combinatorial argument from [1]. In order to shatter m blocks, there must be a codebook that encodes correctly each of the 2^m subsets of those blocks. This requires that the class include at least 2^m distinct codebooks. Since the class of k-dimensional, N-vector codebooks contains exactly

$$\left(\begin{array}{c} 2^k \\ N \end{array} \right)$$

distinct codebooks, the class can shatter no more than

$$\log_2 \left(\begin{array}{c} 2^k \\ N \end{array} \right) \leq kN$$

blocks, so this is an upper bound on the VC-dimension of the class.

With a grayscale codebook, the combinatorial bound above increases sharply. Assuming 8 bits of intensity resolution in our grayscale images, the upper bound on the number of distinct codebooks will be

$$\left(\begin{array}{c} (2^8)^k \\ N \end{array} \right),$$

bounding the VC-dimension as being less than

$$\log_2 \left(\begin{array}{c} 256^k \\ N \end{array} \right) \leq 8kN.$$

There are slightly tighter upper bounds that we can derive for specific tolerance models, but their derivation is complex, and the improvements over the above bounds are minor. For the remainder of this paper, we will simply use the combinatorial bounds described above.

We have now framed a vector quantizer as a pattern classifier, and have determined numerical bounds on the complexity of such a classifier. This allows us to achieve our goal of bounding the r-tolerance distortion on an entire image S as a function of r-tolerance distortion on a small training set S^m drawn from S. By appropriate modification of Equation 3, we have

$$\epsilon_r(c,S) \leq \quad \epsilon_r(c,S^m) + q_r \quad \left(1 + \sqrt{1 + \frac{2\epsilon_r(c,S^m)}{q_r}}\right), \tag{6}$$

$$\text{where } q_r = \frac{d(N,k,r)}{2m}(\ln \frac{2m}{d(N,k,r)} + 1) - \frac{\ln \delta}{2m}.$$

2.5 Relating tolerance to mean distortion rate

In terms of predicting coded image quality, neither the r-tolerance nor (r,t)-tolerance measures appear to be very robust or useful measures. Their main utility is in giving us a starting point from which to look at more commonly used distortion measures. For binary images, a common measure of distortion in vector quantization is average bit error; for grayscale images, the most common measure of distortion is the mean-squared error (MSE). Below, we demonstrate that we can use the above equations to bound the average bit error of a binary codebook given either its average bit error over a training set or the r-tolerance training errors for $0 \leq r < k$. Similarly, we can bound the average MSE of a grayscale codebook given either its MSE over a training set or the (r,t)-tolerance training errors for $0 \leq r < k$ and $0 \leq t \leq 255$.

2.5.1 Binary images

We begin, as always, by considering the case for binary images, deriving a bound on the average bit error. Let

$\epsilon_D(c, S)$ = average bit error of codebook c on image S
$\epsilon_D(c, S^m)$ = average bit error of codebook c on m example blocks drawn from image S.

If an image is broken up into k-dimensional blocks, distortion can be expressed as an average of the r-tolerance bit errors over the range $0 \leq r < k$.

$$\begin{aligned}
\epsilon_D(c, S) &= \text{distortion} = \text{expected bit errors / total number of bits} \\
&= \text{expected bit error per vector / size of vector} \\
&= \frac{1}{k} \sum_{r=1}^{k} r \cdot (\epsilon_{r-1}(c, S) - \epsilon_r(c, S)) \\
&= \frac{1}{k} \sum_{r=0}^{k-1} \epsilon_r(c, S).
\end{aligned}$$

Note that the final sum is over 0 to $k-1$ because $\epsilon_k(c, S) = 0$.

We can then express the average bit error bound as a function of r-tolerance training error using Equation 6:

$$\begin{aligned}
\epsilon_D(c, S) &\leq \frac{1}{k} \sum_{r=0}^{k-1} \left(\epsilon_r(c, S^m) + q_r \left(1 + \sqrt{1 + \frac{2\epsilon_r(c, S^m)}{q_r}} \right) \right) \\
&\leq \epsilon_D(c, S^m) + \frac{1}{k} \sum_{r=0}^{k-1} q_r \left(1 + \sqrt{1 + \frac{2\epsilon_r(c, S^m)}{q_r}} \right), \qquad (7) \\
&\qquad \text{where } q_r = \frac{d(N, k, r)}{2m} (\ln \frac{2m}{d(N, k, r)} + 1) - \frac{\ln \delta}{2m}.
\end{aligned}$$

If we have information about the tolerance errors of our codebook on the training set, we can use the above inequality directly. If we only have information about the mean-squared error on the training set, we must make a few approximations and settle for a looser bound. We define $d(N, k) = \max\{d(N, k, r) : 0 \leq r < k\}$. If $m > d(N, k)/2$ we may substitute this for the dimensions of each r-tolerance model. Then, since $\epsilon_D(c, S^m) \geq \max\{\epsilon_r(c, S^m) : 0 < r \leq k\}$, we can write

$$\epsilon_D(c, S) \leq \epsilon_D(c, S^m) + q \left(1 + \sqrt{1 + \frac{2\epsilon_D(c, S^m)}{q}} \right), \; q = \frac{d(N, k)}{2m} (\ln \frac{2m}{d(N, k)} + 1) - \frac{\ln \delta}{2m}. \qquad (8)$$

2.5.2 Grayscale images

For grayscale images, the transition from tolerance to a more useful distortion measure is not as simple as it is for binary images. Consider the case for MSE distortion: in addition to summing over all tolerances r, we sum over all thresholds t, and must take into account the fact that the pixel errors are squared before being summed. We will follow the convention of the image compression community here by treating the pixel value as an (8-bit) integer rather than as a fraction. This results in individual pixel distortions that range from 0 to 65025 (255^2), rather than 0 to 1.

Much as in the case of the binary images using Hamming distortion, many of the terms in the grayscale MSE distortion expression telescope to give the relatively compact equation

$$\epsilon_D(c,S) \quad = \quad \frac{1}{k}\sum_{t=0}^{255}\sum_{r=0}^{k-1} t^2 \cdot (\epsilon_{r,t-1}(c,S) - \epsilon_{r,t}(c,S)).$$

As in the binary case, substitution may be made into the bounding equations, but the resulting equation is somewhat messy, so we do not detail it here.

3 Empirical results

The results derived in the previous section would have significant value in their own right if they proved to describe the typical behavior of a VQ codebook. However, as we shall see, the theoretical worst-case bounds appear to be far from the typically observed performance, and even far from an "empirical worst-case" derived experimentally. In this section we explore empirically the behavior of VQ codebooks trained on small training sets, in order to compare it with our derived theoretical predictions. Below, we first describe the methodology followed for our series of vector quantizer experiments, and then describe the result of running these experiments on a set of "typical" images. As these results indicate that the bounds of the previous section are much too loose to apply in the average case, we then describe the results of our search for an empirical "worst case" image; one that will allow us to define an empirical upper bound on the difference between the training and test performance of a VQ codebook.

3.1 Methodology

Given a source image S, we first "block" the image into square pixel blocks of k pixels (partial blocks at the edges of the image were discarded). We denote the total number of blocks in the image as M.

From these M blocks, we select a training set S^m of m blocks by random sampling with replacement. This training set is used as input to a program running the Generalized Lloyd Algorithm (described in [12]). The output of the program is a codebook with (locally) minimal distortion on the training set. We denote this codebook as $c(S^m)$.

We then measure $\epsilon_D(c(S^m), S^m)$, the distortion that this codebook imposes on the training set (i.e. the training error) and $\epsilon_D(c(S^m), S)$, the distortion that the codebook imposes on the entire source image S (the test error).

In order to determine the dependence of this value on m, we repeated the above procedure, with fixed k and N for values of m ranging from as small as 50 blocks up to M, the size of the source image. For each value of m, we ran a number of trials, ranging from 50 up to 500 trials, depending on the variance of the observed differences. We denote the expected training and test distortion of a codebook trained on random training set S^m as follows:

$\overline{E}_D(c(S^m), S^m)$ = the expected distortion of $c(S^m)$ on its training set S^m
$\overline{E}_D(c(S^m), S)$ = the expected distortion of $c(S^m)$ on the entire test image S.

We estimate these expectancies, as well as their differences, by calculating the mean and variance of the values over our experimental trials. Our confidence in these values is indicated by the standard error of the mean. Figure 1 illustrates the results of a typical run.

Note that in these experiments, S^m is drawn with replacement and the training and test sets overlap. The results of the previous section are based on the assumption that sampling is done from a distribution, and corroborating these results requires that this assumption not be violated. Many parts of the data compression community, however, call for separate training and test sets, which are generally disjoint and selected *without* replacement from a set of images.

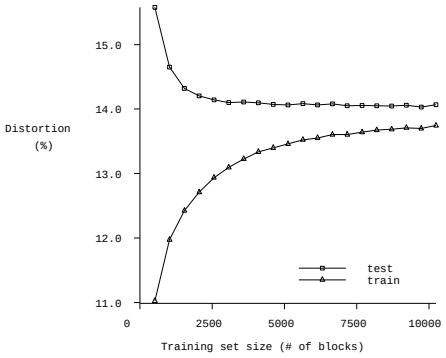

Figure 1: A typical run, this one using a codebook of the 128 25-dimensional vectors on the error-diffused "man" image.

We can reconcile these differences. The difference in distortion between a test set that is disjoint from its training set and one that is not may be described as

$$\epsilon(c(S^m), S - S^m) = \frac{M \cdot \epsilon(c(S^m), S) - m \cdot \epsilon(c(S^m), S^m)}{M - m},$$

where $\epsilon(c(S^m), S^m)$ is the training error of $c(S^m)$, $\epsilon(c(S^m), S)$ is the test error over the whole image, and $S - S^m$ is the "disjoint" test set. Given the size of the image, we can bound the difference between $\epsilon(c, S)$ and $\epsilon(c, S')$, that is, the error on the entire image and on the image minus the training set. In practice, this difference appears to be small, and, for a fixed training set size, goes to zero as the image size increases.

Although the issue of sampling with or without replacement is not as simple, an equivalence, albeit an empirical one, may still be observed. When drawing from a distribution in a continuous domain, the issue of replacement vs. non-replacement is normally moot. Unless the distribution is discrete in some places, the probability of drawing the same point twice is zero. However, when our source is a finite image, there is a non-zero probability that by drawing blocks at random, we may draw the same block more than once, giving us redundancies in the data set.[1]

When the source of our training examples is large, drawing with and without replacement are essentially equivalent; differences appear only when our sample (the training set) takes up an appreciable fraction of the available examples.

Empirically, we have found a relationship between the two sampling paradigms when examining the difference between training and test errors. Specifically, we find that, if S^m is a sample drawn with replacement, and $\tilde{S}^m$ is a sample drawn without replacement,

$$(\epsilon_D(c(\tilde{S}^m), S) - \epsilon_D(c(\tilde{S}^m), \tilde{S}^m)) \approx \frac{M - m}{M - 1}(\epsilon_D(c(S^m), S) - \epsilon_D(c(S^m), S^m)).$$

That is, the $test - training$ distortion differs by a factor of $(M - m)/(M - 1)$. It is not difficult to find extreme cases where this relationship breaks down, but it holds well for all "typical" cases we have examined, and allows conversion between the non-replacement approach and the theoretically examined approach of sampling with replacement (for a more detailed treatment of this relationship, see [3]).

[1] It is easy to confuse this issue with that of drawing distinct but identical blocks. To clarify, imagine that each block in an image is labeled by the row and column of the image in which it appears. When drawing without replacement we could still expect to draw many blocks consisting of the same vector, but we would never draw twice from the same coordinate.

3.2 Average-case experiments

For the single-image learning experiments, we examined both binary and grayscale images from three sources: photographic images from the USC database, MRI brain scans, and computer-generated line drawings. The binary images were generated by halftoning the grayscale images with ordered dithering ([14, 15]) and error diffusion ([6]).

For each of the images tested, the difference between the observed behavior and the worst-case bound was remarkable. The true distortion of the codebook on the test image rapidly approaches its asymptotic value, while the theoretical bound on the codebook's distortion remains surprisingly high, even for large training set sizes. In Figure 2, we plot the bound from Equation 8 using an upper bound on the VC-dimension of the codebooks involved.[2] We also plot the bound assuming a trivial lower bound VC-dimension of 128 (assuming each codebook vector covers only a single vector). Unfortunately, the graph indicates that even this bound, using the trivially small VC-dimension, is too conservative to provide realistic guidance for a typical problem. Even using the tighter "tolerance" bound of Equation 7 along with the tolerance information gleaned from the training process fails to produce an upper bound on distortion that approaches observed behavior.

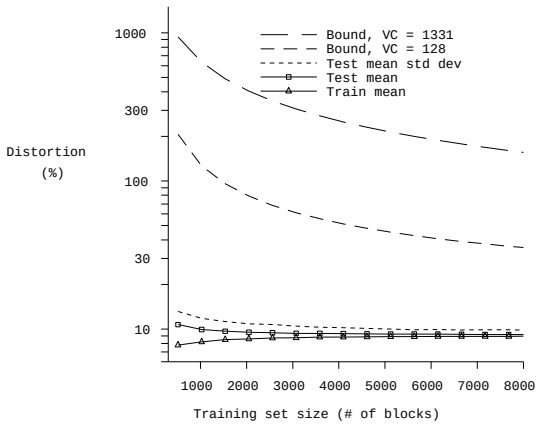

Figure 2: The theoretical upper bounds on test distortion of a codebook of 128 16-dimensional binary vectors trained on a typical image. The confidence parameter δ in this case is 0.5, indicating that the bound is guaranteed to hold at least 50% of the time. Two bounds are plotted: one using the upper bound on the possible VC-dimension of the codebook (1331), and one using a trivial lower bound on the VC-dimension of the codebook (128). The disparity between even the lower of these two bounds and the actual test distortion (also plotted) indicates that the worst-case bounds may not provide useful information. Image used was error-diffused "man," from the USC database.

Having determined that the formal bound does not provide direct guidance in the typical case, we now empirically examine the functional form of this case in an attempt to derive some experimental guidance.

It proves instructive to look at Equations 7 and 8 as placing an upper bound on the quantity $(\epsilon_D(c(S^m), S) - \epsilon_D(c(S^m), S^m))$ with confidence at least $1 - \delta$. This is a bound on how well the codebook's performance on the training data will indicate its performance on the entire image. Below, we examine this value empirically as it changes with training set size (m), block size (k), and codebook size (N) for typical images.

[2] The value we use is $\lfloor \log_2 \left(\begin{array}{c} 2^{16} \\ 128 \end{array} \right) \rfloor = 1331$.

We determined that the value of $(test - train)$ distortion closely follows the first-order polynomial:

$$(test - train) = \frac{\alpha}{m + \beta}. \tag{9}$$

For both the binary and grayscale images, there is a remarkably good fit to Equation 9 when sampling is done with replacement (see Figure 3). The only noticeable deviations from the polynomial model are for small training set sizes or when large codebooks are used. This is consistent with observations made in [18], pointing out that when a learner is sufficiently powerful and a training set is sufficiently small, rather than learning to generalize, the learner simply memorizes data to some extent. Memorization is a qualitatively different phenomenon from generalization, and is beyond the scope of this study.

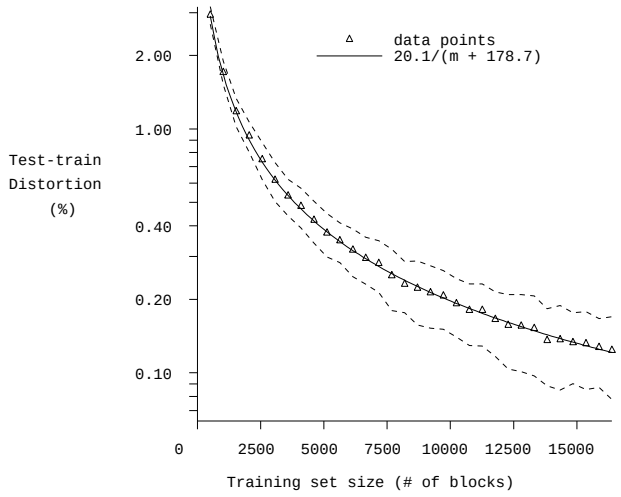

Figure 3: The best fit of a typical run sampled with replacement to an inverse first-order polynomial. Dashed line represents standard error of the mean. The image used was error-diffused "man," quantized with a codebook of 128 16-dimensional vectors.

When S^m is drawn without replacement, the $(test - train)$ distortion again follows the inverse first-order polynomial, but with a modifying linear factor. Namely,

$$(test - train) = \frac{M - m}{M - 1} \cdot \frac{\alpha}{m + \beta}. \tag{10}$$

Figure 4 plots the generalization curves of codebooks sampled both with and without replacement, as well as their best fit to their respective equations.

3.2.1 Learning complexity for "typical" images

One surprising observation from our experiments is that, in spite of the fact that training and test distortion varied widely from image to image, their difference was relatively constant, given a fixed training set size, block size and codebook size (see Figure 5). All of the photographs used from the USC database fit Equation 9 with similar values of α, and thus have similar learning complexities. Images from the MRI brain scans and

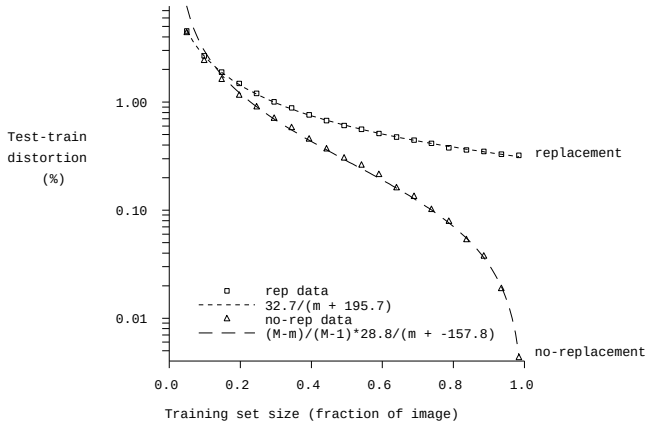

Figure 4: Best fits of runs sampled with and without replacement to their respective models. Image quantized was error-diffused "man," using a codebook of 128 25-dimensional vectors.

the computer-generated line drawings gave appreciably different and lower learning complexities than did the photographs.

It is instructive here to give these values a concrete example: if we have a binary VQ for which $\alpha = 25$, then to achieve an expected $(test - train)$ distortion of less than 0.1%, one must train on at least $\alpha/0.001$, or 2500 binary blocks.

It appears, based on these experiments, that one may quantify a learning complexity that is "typical" for a class of images, for example, photographs, satellite images, line drawings, etc. Below, we describe how we have quantified α, the learning complexity, for the class of binary and grayscale photographs. Although β is technically also a factor in the equation, empirically it seems to have a small value, and thus does not play a major role in the equation.

Figure 6 plots learning complexity α for binary and grayscale images as it varies with block size and codebook size. Note that for the grayscale images, the value of α is not significantly dependent on block size, but only on the size of the codebook. For the binary images, there is a small but significant increase in α with increasing block size.

Defining $n = log_2 N$ for brevity, we can extrapolate the following "characteristic" equations:

$\alpha_{binary} = 0.75 \cdot 2^{0.0075kn+0.50n+0.017k}$
$\alpha_{grayscale} = 23600 \cdot 2^{0.79n}$.

3.3 Learning complexity for "worst-case" images

The learning complexity is not an indication of how difficult it is to quantize an image well, but of how much of the image we need to see to *know* how well we can quantize it. For a given algorithm, there must be some image, or set of images $S^*_{N,k}$ for which the test and training distortion of a codebook with N k-dimensional vectors will converge most slowly on the average. Being able to describe and produce such a "worst-case" image would have great benefits. Measuring the generalization curve on this image would provide an upper bound on learning complexity, an α^* such that, for an arbitrary image, given training set S^m, the true distortion of a codebook would be less than or equal to $\epsilon_D(c(S^m), S^m) + \alpha^*/m$ with high probability. We will refer to this α^* as the maximum learning complexity of the codebook (without reference to an image).

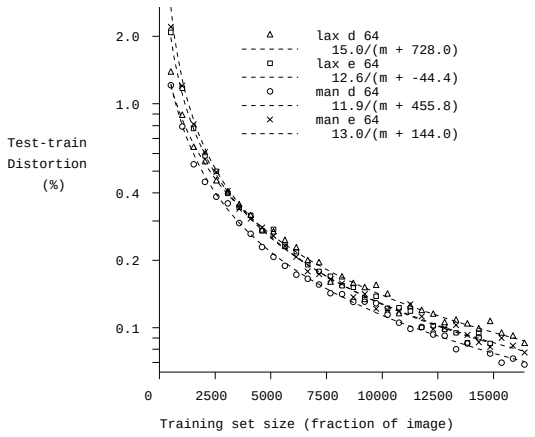

Figure 5: The rate of convergence of test and training distortion was roughly the same for all halftoned photographs, despite widely varying individual test and training distortions.

We have been unable to analytically derive such an α^*, but have determined that for the grayscale case, the learning complexity of an image appears to be related in a near-linear manner to the entropy of the image. For binary images, this relationship appears to be non-monotonic; α is at a maximum for images of intermediate entropy (see Figure 7). Although the binary case needs more study, it is straightforward to construct a grayscale image that, following the entropy plot, should have maximal learning complexity: we construct an completely random image (i.e. one with maximal entropy), in which each pixel is either completely on or completely off. Note that even though this image is now technically binary, it still qualifies as a (very high-contrast) grayscale image. Figure 8 plots the empirical "worst-case" learning complexities for this image. In this case, there appears to be some dependence on block size k, but it is small enough that it may be ignored in this first-order approximation. The empirical "worst-case" α may be approximated as

$\alpha_{wc} = 286000 \cdot 2^{0.634n}$.

It may be noted that even compared to this "worst-case," an order of magnitude worse than the average observed case, the theoretical bounds of the previous section are still too loose to provide useful guidance.

4 Multiple-Image Learning Experiments

While for some applications, a codebook is designed solely for the purpose of encoding a particular image, it is more common for one codebook to be trained on a *set* of images, for the purpose of encoding a different set of images. In this section, we discuss the extension of results from the previous section to the problem of learning and encoding multiple images.

There are a few important differences that must be considered when working with multiple images, or when the test images are different from the source of the training examples. Most obviously, there is the problem that the difference between the training and test distortions will no longer asymptotically converge to zero. Second, even if we *could* guarantee some bound on the difference between the training distortion and test distortion, we would have no guarantee of performance on individual test images. A guarantee that $(test - train)$ over a set of ten test images was less than 5% could still give 1% distortion over nine of the images and 40% distortion on the tenth. Although the formal theory described in Section 2 is stymied by the

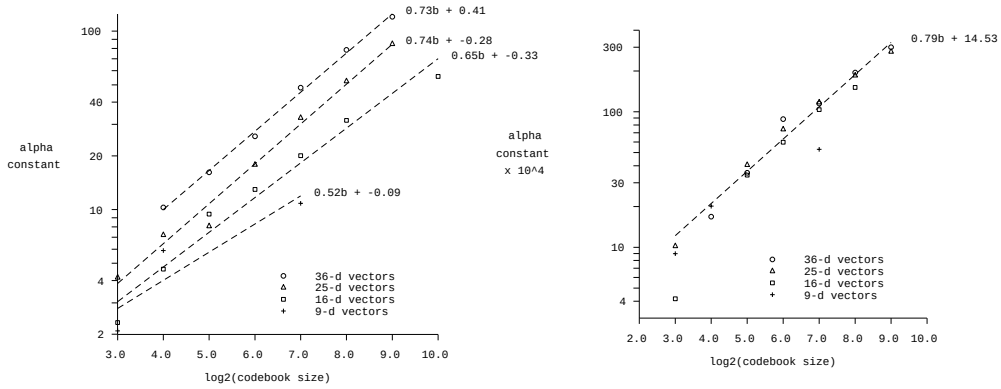

Figure 6: Learning complexity (α) as a function of vector dimension and codebook size for binary (left) and grayscale (right) images. The image used in both cases was "man," which was halftoned for the binary case by error-diffusion.

use of disjoint training and test images, we have found that the practical performance examined in Section 3 may still be used with little modification. We detail this below.

If a codebook is trained on an image set S and tested on an image set T, its training distortion $\epsilon_D(c(S^m), S^m)$ will approach $\epsilon_D(c(S), S)$ as the training set size increases, while the test distortion $\epsilon_D(c(S^m), T)$ will approach its limit of $\epsilon_D(c(S), T)$. Since typically $\epsilon_D(c(S), S) \neq \epsilon_D(c(S), T)$, the test and training distortion will not converge, and thus cannot be fit by an asymptotically converging form like Equations 9 and 10.

We thus examine the first derivative of Equation 9,

$$\Delta(test - train) = \frac{d}{dm}(test - train) = -\frac{\alpha}{(m+\beta)^2}, \tag{11}$$

which indicates by how much an additional training block should improve our performance. Regardless of the asymptotic error rates $\epsilon_D(c(S), T)$ and $\epsilon_D(c(S), S)$, this value will approach zero as m increases, indicating that further increasing the training set size will have little effect.

Because our previous single-image generalization curves fit Equation 9, they will also fit Equation 11 with the same parameter values. We now see how these values must be adjusted to accommodate multiple images. For these experiments we chose two image sources: a set of seven photgraphs from the USC database and a set of eleven GOES weather satellite images.

In the first series of experiments with these images, we compared the derived values of α for single-image training sources (as from the previous section) with sources comprised of many images. As illustrated in Figure 9, the learning complexities of the multi-image sources was almost the same as that of the single image sources, indicating a relative unimportance of the "size" of the training image source.

We then examined the effect on α of testing on images that were not part of the training set. In these experiments, a codebook was trained on a set of either ten of the GOES images (reserving one day), or on six of the USC images (reserving "man"). Each codebook was then tested on an image in its training set, the reserved image from the training image source, and the reserved image from the *other* image source. The results are plotted in Figure 10. The generalization curves *do* match Equation 11 well, and indicate that the convergence rate, based on α, is relatively insensitive to the source of the test images.

The implications of these observations are important. They indicate that, to a degree, the empirical results of Section 3 are applicable to multiple-image learning as well. That is, if we are training on a source

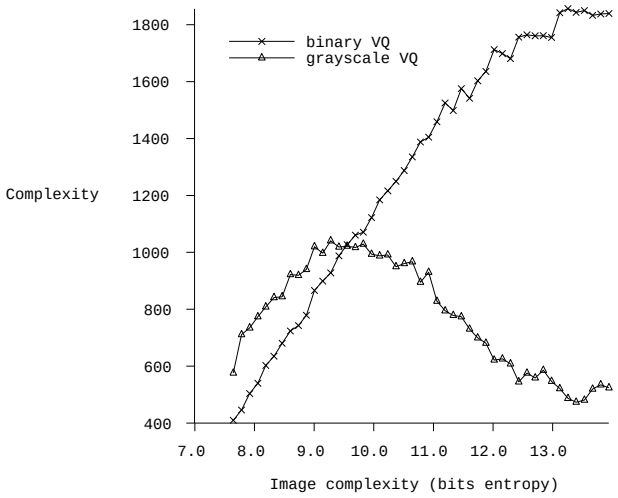

Figure 7: Learning complexity as a function of random image entropy. The plot for the binary case has been normalized to fit on the same scale as that of the grayscale plot.

of image blocks S (say, a fixed finite set of of weather satellite images), in an attempt to encode blocks from another source T (say, the infinite set of *all* weather satellite photos we will see in the future), then there is some minimum distortion which is possible if we use all the available data: $\epsilon_D(c(S), T)$. Our results indicate that the rate at which this minimum is approached is relatively independent of the "size" of sources S and T. This does *not* indicate that it makes no difference on what data we train our codebook; it simply suggests that (for non-degenerate cases), we will approach asymptotic performance at the same rate, regardless of our training and test image sources.

In all of the tested cases, the approach to the asymptotic performance was governed by the learning complexity of the codebook, with little variation in α as different training and test images were used. This indicates that, to a degree, the convergence results described in the previous section are characteristic of the training algorithm, and are independent of the images involved.

5 Discussion

In this last section, we first discuss the theoretical implications of the results in the previous two sections. We then discuss the practical implications of these results, and suggest guidelines that users of VQ techniques may use to select appropriate training set sizes. We then conclude by briefly recapitulating the results of this paper and suggesting directions for future research.

5.1 Implications of empirical results for theory

The bounds derived in Section 2 are worst-case bounds, which hold regardless of the input distribution and codebook-design algorithm. It has been shown in [9] that under certain circumstances, much tighter bounds may be derived if something is known about the design algorithm. Consider the case of an arbitrary classifier that can achieve zero training error on m training examples. Using Equation 3, we can bound its

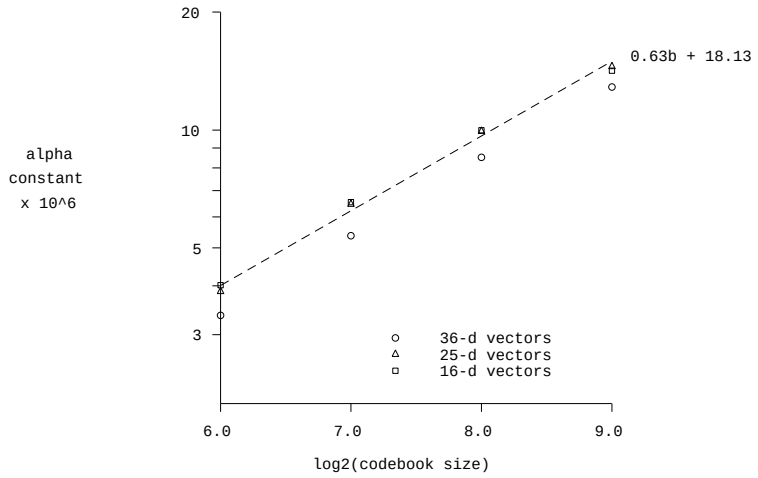

Figure 8: Dependence of α on codebook size for an empirical "worst case" image.

generalization error as:

$$\epsilon(c,(t,\mathcal{P})) \leq \frac{d}{m}(\ln\frac{2m}{d}+1) - \frac{\ln\delta}{m}.$$

If, however, we know that the classifier follows a Bayes-optimal decision rule, then we can show that the expected worst-case behavior is no greater than

$$\epsilon(c,(t,\mathcal{P})) \leq \frac{d}{m}.[9]$$

Empirical studies in [2] have shown that even for some very simple zero-training-error learning problems, expected generalization may be close to the latter, Bayes-optimal bound.

No such tighter bound is known for the case when training errors are non-zero, and we know nothing about the optimality, Bayes or otherwise, of the GLA used to design codebooks in these experiments. However, the simplicity of the observed $(test-train)$ curves suggests that the non-zero training error bounds may be of a similar form, such as in Equation 9.

This d/m "inverse power law" has been observed in fields other than formal learning theory as well. Similar convergence to asymptotic performance is seen in the expected parameter mismatch of dynamical systems as a function of their training set size [13].

5.2 Implications of empirical results for practice

Given perfect knowledge of some source, there is some minimum test distortion of that source which a codebook design algorithm can achieve for a fixed bit rate. If our source is a fixed image, or fixed set of images S, then this minimum is $\epsilon_D(c(S),S)$. To ensure that a codebook is within ξ of the minimum, it is sufficient to ensure that its test distortion is within ξ of the training distortion. Since $(test-train) \approx \alpha/m$, it should be sufficient to train on α/ξ blocks to achieve this performance.

Another use for these results is relating a limited codebook training time to $(test-train)$. The time to train using the GLA is proportional to both training set size and codebook size. For example, on a

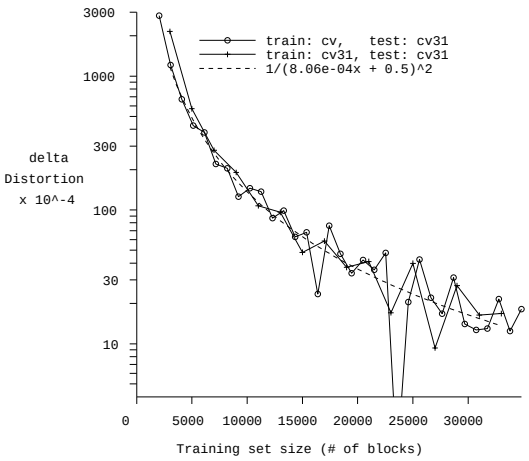

Figure 9: The rate of convergence of test and training distortion was roughly independent of the size of the source from which the training set was drawn. Codebooks trained on blocks drawn from the full set of 10 GOES images (cv) converge to its asymptote at almost the same rate that a codebook trained on only a single GOES image (cv31) does.

DECstation 5000, with 16-dimensional codebook vectors, the training time for our implementation required approximately 50μs per codebook vector per training element. With this knowledge, we can substitute training time for training set size (with an appropriate constant factor). Then, the derived learning complexity can give guidance as to how close to optimum one can come for a fixed training algorithm given a fixed amount of training time.

Thus, given the bounding learning complexity for a set of images, we can make useful decisions about appropriate training set sizes and training times for a VQ problem. In Section 3, we derived a parameterized equation for α using a set of images that maximized its value over the source images we had available. Using this value of α may prove useful to other practitioners, or, with a little experimentation, they may derive values of α appropriate to their own problem domains. In some domains, with very regular or very noisy images, the learning complexity may be significantly smaller, and further computational savings may be realized by basing one's training set sizes on this smaller value. Practitioners using VQ for other applications, e.g. for compressing speech or sonar data, may likewise find it useful to perform their own experiments to establish whether our results and guidelines generalize to their areas of interest.

If a practitioner wishes to derive the learning complexity of his or her own data set for a particular block size k and bit rate b, they may do it by estimating $\alpha(k,b)$ for a single (small) value of m. Using the chosen settings, a practitioner may then repeatedly extract m blocks from his or her training set, train on them, test on another randomly extracted set of blocks and determine $(test - train)$ for that trial. By repeating this procedure and averaging, one may derive an increasingly accurate estimate of the $E[test - train]$ for that value of k, b, and m. The learning complexity $\alpha(k,b)$ is then simply $E[test - train]/m$. For example, let us say that we want to compress a set of MRI brain scans, designing a separate codebook for each (to minimize distortion). Let us say that the image has a resolution of 512×512 pixels, or 16,384 4×4 blocks. As noted in Section 3, there is some residual lack of fit to the α/m model at very small training set sizes, so we run our experiments using an initial training set size of $m = 3000$ blocks, sampled with replacement. The mean training and test errors are 5.92% and 6.58% respectively, giving an empirical mean $(test - train)$ distortion of 0.67% with a standard error of 0.11%. This gives us an estimated learning complexity of $\alpha = 20.1$, with

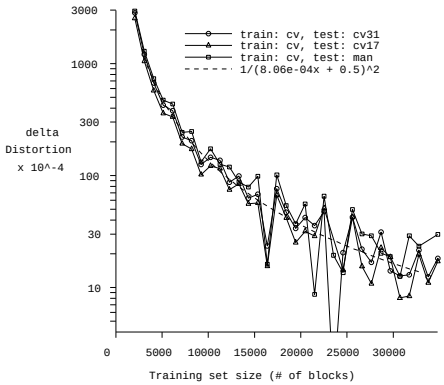

Figure 10: Plot of $\Delta(test - train)$ for codebooks were designed by drawing random blocks from 10 of the GOES images. These were then tested on one of the images in their training set ("cv31"), one image not in their training set ("cv17"), and one USC image ("man").

23.4 and 16.8 as upper and lower limits for the expected value of α. Now, let us say that we want our codebook to have distortion within 0.1% of the best we can do with the given training algorithm and values of k and b. This represents a tiny fraction of the approximately 6% distortion that we expect our final codebook to produce. To achieve this performance we need to choose a training set large enough that the expected value of $(test - train)$ is less than 0.1%. If we are sampling without replacement, then solving $(test - train) = (M - m)/(M - 1) \cdot \alpha/m$ for m gives a suggested training set size of $m = 9636$. By running a series of experiments with this training set size, we find an average $(test - train)$ difference of 0.082%, confirming our expectations.

With a bit more work, we may extrapolate over different block sizes and bit rates as well as different training set sizes. The observed form of the learning complexity of a codebook is approximately $\alpha = \gamma_1 \cdot 2^{\gamma_2 kn - \gamma_3 k - \gamma_4 n}$. If the block size is fixed, then this exponent is linear in the bit rate. Similarly, for a fixed codebook size, the exponent is linear in the block size. Since the exponent of the empirically observed form is linear in both k and n, determining the appropriate coefficients is a matter of determining $\alpha(k, n)$ for two distinct values of k or n (each of which may be done at a single small value of m). By performing a linear fit to the logarithm of these derived $\alpha(k, n)$, we may derive generalized equations like those in Section 3.2.1.

5.3 Summary and future directions

In this paper we have examined how the performance of a memoryless vector quantizer changes as a function of its training set size. Specifically, we studied how well the training set distortion predicts test distortion when the training set is a randomly drawn subset of blocks from the test or training image(s). We have demonstrated formal upper bounds on $(test - train)$ distortion as a function of vector dimension, bit rate, and training set size. These bounds turn out to be much too loose to be of practical help, so we have demonstrated practical guidelines derived empirically for a range of test images. These guidelines depend only on codebook size and dimension, and appear to be somewhat robust across classes of images. There is, however, a great deal of work that remains to be done in this area. A few of the most obvious directions are:

1. More inquiry is necessary into α^*, the "worst-case" learning complexity. As in Section 3.3, by "worst case" we do not mean producing the greatest minimum distortion, but rather, requiring the largest training set size on the average, to achieve less than some specified difference in test and training distortions. The formal bounds here are algorithm independent, and do not take into account the

limited learning ability of GLA training. Therefore, the worst-case upper bound will be too high. The simulations described in this paper demonstrate empirical worst-case lower bounds, but because the images used were selected by arbitrary criteria, there almost assuredly exist distributions that are more difficult to learn than the ones we have examined. Working from the mathematical framework of the covering problem and the r-tolerance model, it may be possible to formally prove that a "worst-case" distribution exists, and to then measure generalization, analytically or empirically, on that distribution. This would provide an exact bound on worst-case learning complexity.

2. We have completely ignored the implications of this work for achieving minimum distortion for a fixed bit rate (or minimum bit rate for fixed distortion), given a fixed allowable training time. Recent work described in [11] and elsewhere, addresses the problem of minimizing training distortion. From a bound on training distortion, with the work here bounding the difference $(test - train)$, it should be possible to directly bound the test distortion of an image as a function of its codebook training set size.

3. Finally, we have only touched on the problem of identifying "classes" of images for training and testing codebooks. Further work in this area is needed to determine over what set of images a given learning complexity is valid.

Despite the amount of research that still needs to be done in this area, the work described in this paper has made one thing clear: it is possible to design good codebooks using only a fraction of the computational power typically used by the normal "exhaustive" training paradigm.

Acknowledgments

Portions of this research were funded by National Science Foundation grant numbers CCR-9108314 and MIP-9110508 and Hewlett-Packard Laboratories. We would also like to thank Professor Les Atlas of the University of Washington Department of Electrical Engineering and Professor David Madigan of the University of Washington Department of Statistics for many helpful discussions and suggestions during the preparation of this paper. A portion of this work was done while D. Cohn was at the Department of Computer Science, University of Oregon.

References

[1] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, October 1989.

[2] D. Cohn and G. Tesauro. How tight are the Vapnik-Chervonenkis bounds? *Neural Computation*, 4(2):249–270, March 1992.

[3] D. Cohn. Separating formal bounds from practical performance in learning systems. Ph.D. dissertation, Department of Computer Science and Engineering, University of Washington, 1992.

[4] P. Cosman, K. Perlmutter, S. Perlmutter, R. A. Olshen, and R. M. Gray. Training sequence size and vector quantizer performance. In *Proceedings of 25th Asilomar conference on Signals, Systems, and Computers*, Asilomar, CA, November 1991, pages 434–438.

[5] J. Crutchfield and K. Young. *Computation at the onset of chaos*, pages 223–269. Addison-Wesley, 1990.

[6] R. Floyd and L. Steinberg. An adaptive algorithm for spatial grey scale. *SID Int. Sym. Digest of Tech. Papers*, pages 36–37, 1975.

[7] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, 1992.

[8] R. M. Gray. Vector quantization. *IEEE ASSP Magazine*, 1:4–29, April 1984.

[9] D. Haussler, M. Kearns, and R. Schapire. Unifying bounds on the sample complexity of Bayesian learning theory using information theory and the VC dimension. In *Proceedings of the 4th Annual Workshop on Computational Learning Theory*, pages 61–74, San Mateo, CA, 1991. Morgan Kaufmann.

[10] F. Itakura and S. Saito. Analysis synthesis telephony based on the maximum likelihood method. In *Proceedings of the 6th International Congress on Acoustics*, 1968.

[11] J. Lin and J. Vitter. ϵ-approximations with minimum constraint violation. Unpublished manuscript, November 1991.

[12] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–95, January 1980.

[13] L. Ljung. *System Identification – Theory for the User.* Prentice Hall, Englewood Cliffs, NJ, 1987.

[14] A. N. Netravali and B. G. Haskell. *Digital Pictures Representation and Compression.* Plenum Press, New York and London, 1988.

[15] R. Ulichney. *Digital halftoning.* MIT Press, Cambridge, MA, 1987.

[16] V. Vapnik. *Estimation of dependencies based on empirical data.* Springer-Verlag, New York, 1982.

[17] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[18] S. Weiss and C. Kulikowski. *Computer Systems that Learn.* Morgan Kaufmann, San Mateo, CA, 1991.

[19] P. Zehna. *Probability Distributions and Statistics.* Allyn and Bacon, Boston, 1970. pp. 286, 289.