Error Modeling for Hierarchical Lossless Image Compression

Paul G. Howard and *Jeffrey Scott Vitter*

A shorter version of this paper appears in *Proceedings of the IEEE Data Compression Conference*, Snowbird, Utah, March 23–26, 1992, 269–278.

Error Modeling for Hierarchical Lossless Image Compression[1]

Paul G. Howard[2]
Visual Communications Research
AT&T Bell Laboratories
Holmdel, N.J. 07733–3030

Jeffrey Scott Vitter[3]
Department of Computer Science
Duke University
Durham, N.C. 27706–0129

Abstract

We present a new method for error modeling applicable to the MLP algorithm for hierarchical lossless image compression. This method, based on a concept called the *variability index*, provides accurate models for pixel prediction errors without requiring explicit transmission of the models. We also use the variability index to show that prediction errors do not always follow the Laplace distribution, as is commonly assumed; replacing the Laplace distribution with a more general symmetric exponential distribution further improves compression. We describe a new compression measurement called *compression gain*, and we give experimental results showing that the MLP method using the variability index technique for error modeling gives significantly more compression gain than other methods in the literature.

Index terms: Data compression, predictive image coding, error modeling.

---

[1] A shorter version of this paper appears in *Proceedings of the IEEE Data Compression Conference*, Snowbird, Utah, March 23–26, 1992, 269–278.

[2] Work was performed while the author was at Brown University and at Duke University. Support was provided in part by NASA Graduate Student Researchers Program grant NGT–50420 and by a National Science Foundation Presidential Young Investigator Award grant with matching funds from IBM. Additional support was provided by a Universities Space Research Association/CESDIS associate membership.

[3] Work was performed in part while the author was at Brown University. Support was provided in part by a National Science Foundation Presidential Young Investigator Award grant with matching funds from IBM and by Air Force Office of Scientific Research grants F49620–92–J–0515 and F49620–94–1–0217. Additional support was provided by a Universities Space Research Association/CESDIS associate membership.

1 Introduction

In this paper we address the question "How well can we hope to compress images without loss of information, disregarding the computational resources required?" We present a computation-intensive method that achieves about 7 percent better compression than the lossless mode of the JPEG proposed standard. The significance of this work is threefold. First, our method can be used in practice when maximum compression is required. Second, as processors become faster and cheaper, our method becomes feasible for everyday use. Third, we provide a standard to which we can compare other faster methods to see how much compression they sacrifice.

In [2] we introduced a paradigm for lossless image compression including four components:

- *Pixel sequence.* Careful selection of the pixel processing order can permit progressive compression, parallel computation, and improved compression efficiency.
- *Prediction.* Accurate prediction of pixel values based on the values of previously coded pixels allows us to encode only the prediction errors (differences between the actual and predicted values), leading to a great saving of code length. Predictive coding is the basis for most lossless image compression techniques.
- *Error modeling.* Precise characterization of the errors inherent in the prediction process gives us a model that can be used effectively by a statistical coder.
- *Coding and computational efficiency.* Arithmetic coding allows us to obtain optimal average code length; Huffman coding and quasi-arithmetic coding, our new variant of arithmetic coding, give almost optimal compression with faster running times.

We also introduced an algorithm for progressive coding based on a hierarchical pixel sequence. This *multi-level progressive* algorithm is called MLP. In this paper we focus on the error modeling component of MLP, and present a practical method of modeling prediction errors that yields substantial improvements in compression efficiency.

In the MLP algorithm, the pixels in an image are divided into levels, each level having twice as many pixels as the preceding one. The pixels in a level are arranged in a (possibly rotated) checkerboard pattern. Within a level we apply three operations to each pixel:

1. We predict the pixel's intensity based on the intensities of other nearby pixels (with known values) in all directions, and compute the prediction error.
2. We find an appropriate model for the prediction error, consisting of a probability distribution specified by a variance and possibly other parameters.
3. We encode the prediction error using the estimated model in conjunction with a statistical coder.

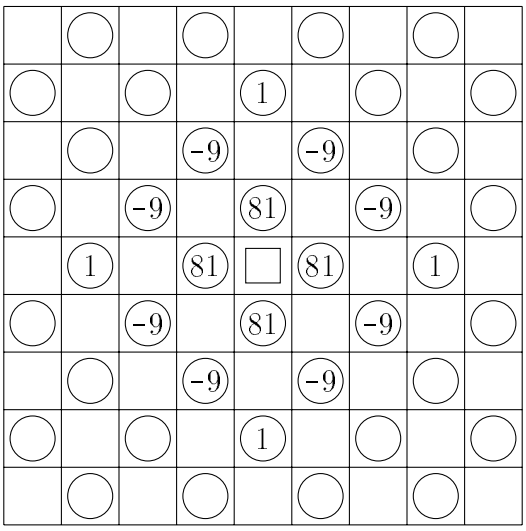

Figure 1: MLP prediction neighborhood. Before MLP processes its last level, the values of the pixels with circles in them are known. The pixels with numbers in their circles are used to predict the value of the pixel at the center, marked with a square. The numbers in the circles are the relative weights given to the predicting pixels; if all 16 points can be used for prediction, the prediction will be the weighted sum of the predicting pixels divided by 256. All unmarked pixels will also be predicted during the last level.

The last level of the process is illustrated in Figure 1.

The best predictions come from linear combinations of the nearest 4, 12, or 16 pixels, the coefficients being chosen to perform polynomial interpolation. Prediction using 16 pixels is shown in Figure 1; the coefficients are those which exactly fit a polynomial of the form

$$\sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij}\, x^i y^j$$

to 16 data points arranged as in the figure.

In passing, we note that the predicting pixels are symmetrically placed, and have only three different weights, namely $w_1 = 81$, $w_2 = -9$, and $w_3 = 1$. Since the weights are normalized, different prediction methods differ only in the ratios $r_{21} = w_2/w_1$ and $r_{32} = w_3/w_2$; in this case the ratios are the same: $r = r_{21} = r_{32} = -1/9$. In practice, we can obtain slightly better compression for many images by using different predictors for different levels. Using $r = 0$ for all but the last two levels (giving 4-point prediction), $r = -0.06$ for the next-to-last level, and $r = -0.12$ for the last level improves compression for most of our test images by about one half percent. Note also that the corner pixels have very low weights and hardly contribute to the prediction, so we can safely omit them. These corrections are somewhat *ad hoc* and do not give a large improvement, so do not include them in our experimental results.

Arithmetic coding can encode the prediction errors optimally with respect to any

given error model. Finding a good model is at least as important for compression efficiency as making good predictions. Using too flat a distribution gives too low a probability (and hence an excessively long code) to small error values, which occur frequently. On the other hand, using too peaked a distribution allocates too little probability to large errors. To keep the code length short, we prefer that the error modeling be done implicitly, with no need to transmit side information about the models used. Of course, the decoder must be able to mirror any error-modeling computations performed by the encoder.

In this paper we introduce the notion of error modeling by *variability index*. In Section 2 we present and motivate the algorithm. In Section 3 we use the variability index as an experimental tool to show that prediction errors are not always Laplace distributed, as is widely assumed. We also indicate a family of distributions that can provide more accurate error models. In Section 4 we describe experiments showing that application of the variability index algorithm to Laplace distributions leads to an improvement in compression efficiency of about 4 percent compared with implementations with explicit transmission of variances; further refinement by allowing a wider family of error distributions gives another half percent improvement. In Section 5 we discuss the direction of our current work.

2 Modeling by Variability Index

In this section we assume that prediction errors are random variates from a Laplace (or symmetric exponential) distribution with zero mean. This assumption is based on considerable experimental evidence beginning with O'Neal [6], although in Section 3 we provide evidence that Laplace distributions are not always the best models for predictive coders.

Our goal is to estimate the *local* variance of the pixel prediction errors in a given level of the MLP encoding. This will allow us to select the appropriate Laplace distribution (the one with the estimated variance) to use in encoding each pixel's prediction error by arithmetic coding. Using a single variance for the entire level (as in [2]) disregards local variations in the image; estimating and explicitly encoding variances for small sections of the image (also done in [2]) incurs considerable overhead. It would seem that previously encountered prediction errors of nearby pixels would be related to a given pixel's error, but there is no obvious way to use them directly.

We estimate the variance by an indirect method: for each pixel we compute a *variability index*, a quantity strongly correlated with the local variance. On the supposition that pixels with similar variability index will have similar error models, we adaptively estimate the local variance based on the variability index. The algorithm is as follows:

1. For each pixel in the current level, we compute the variability index.

2. We sort the pixels in variability index order.

3. We initialize the variance estimate V. The choice of the initial value of V is not

critical—for convenience we use the value it had one tenth of the way through the previous level.

4. For each pixel (in order of decreasing variability index):

   (a) We code the prediction error d, using a distribution with zero as its mean and V as its variance.

   (b) We use the prediction error d to update V. The variance is just the mean squared error; our variance estimate is a weighted mean squared error, computed by exponential smoothing:

   $$V := f \cdot V + (1 - f) \cdot d^2,$$

   where f is a smoothing parameter. Since the prediction error d is coded in step 4(a), both the encoder and the decoder can use its value in step 4(b).

Note that the variability index is used only for sorting the pixels, then discarded. We use decreasing variability index order because compression efficiency is less dependent on accurate variance estimates for large variances; hence we encode pixels with large variances first in each level, before our variance estimate has stabilized. The sorting step removes any natural ordering of the pixels, so both the encoder and the decoder must maintain the pixel coordinates.

We have tried several different quantities for the variability index, including both intensity values and prediction errors of pixel neighborhoods of various sizes. Empirically, the most effective is simply the variance of the four nearest pixels. The smoothing parameter f must also be selected. Experiments show that the best results come from a large value like 0.992; smaller values make the estimated variance too sensitive to random fluctuations in the variability index.

3 Distribution Selection

When we code the pixels of a level in variability index order, we can plot histograms of the error distributions for different values of the variability index. A typical set of plots is shown in Figure 2. Two facts are apparent. First, the error distributions are not always Laplacian; in fact, for large variability index they appear closer to normal. Second, the distributions become closer to the Laplace distribution for smaller values of the variability index.

This leads us to consider a family of generalized symmetric exponential distribution functions which includes both the Laplace distribution and the normal distribution. Distributions in this family have the form

$$f_{n,\sigma}(x) = \frac{\alpha_n}{\sigma} \exp\left(-\beta_n \left|\frac{x}{\sigma}\right|^n\right). \tag{1}$$

From the constraints on a probability distribution with variance σ^2, we can find the values of parameters α_n and β_n:

$$\alpha_n = \frac{n}{2}\left(\frac{, (3/n)}{, (1/n)^3}\right)^{1/2}, \qquad \beta_n = \left(\frac{, (3/n)}{, (1/n)}\right)^{n/2}.$$

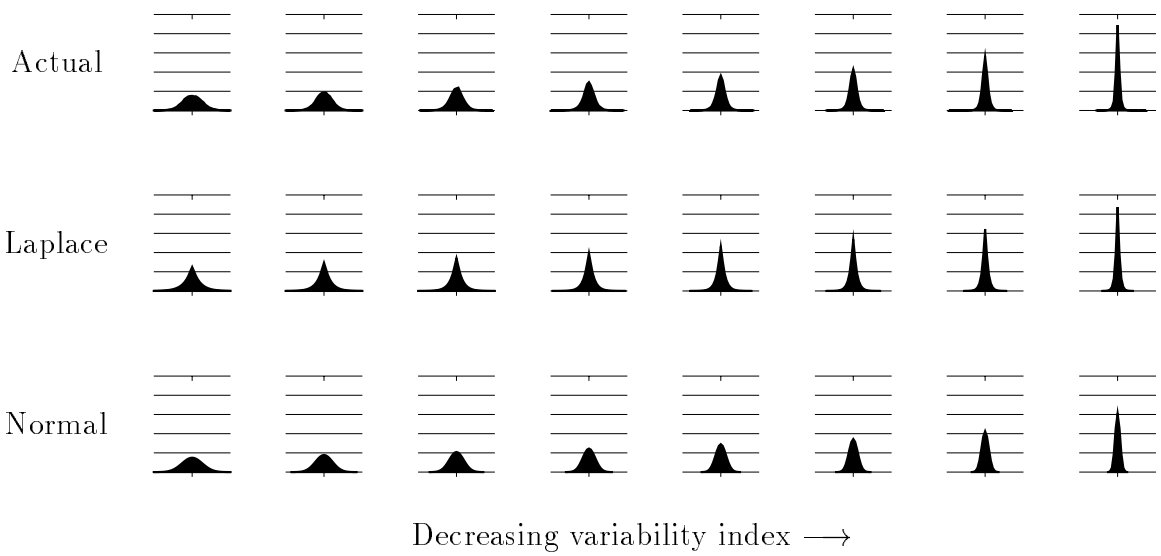

Figure 2: Histograms showing distributions of prediction errors. The top row shows the actual distributions of prediction errors for eight batches of 16,384 points each taken from the last level of MLP's coding of Band 4 of the Donaldsonville Thematic Mapper data set. The variability index decreases from left to right. The second and third rows show the discretized Laplace distributions and normal distributions with the same variances as the actual data in the corresponding column. The horizontal lines are at intervals of 10 percent probability; prediction errors from -20 to $+20$ are plotted.

For the Laplace distribution ($n = 1$), we have $\alpha_1 = 1/\sqrt{2}$ and $\beta_1 = \sqrt{2}$; for the normal distribution ($n = 2$), we have $\alpha_2 = 1/\sqrt{2\pi}$ and $\beta_2 = 1/2$.

We can use these distributions to optimize the exponent n in Equation (1), either image by image or for a class of images. The best values of n for our individual test images range from 1.0 to 1.9, the best overall value being 1.25. For many of our test images we obtain a further slight improvement when we vary the exponent within each level; in our experiments we use a given starting exponent and reduce it linearly toward 1 as we progress through each level. The improvement obtained by using a non-Laplace distribution is about a half percent; the extra improvement obtained by varying the exponent within a level is small, as is the improvement from selecting the optimal exponent image by image.

It appears that the distributions we see are actually mixtures of normal or near-normal distributions. High variability indices can arise only in regions with high local variance, so the mixture distribution contains contributions from only a small range of distributions. Low variability indices, on the other hand, can come either from regions with low local variance or from the not unusual occurrence of small deviates in regions with high local variance. The resulting mixture has contributions from many different distributions, and tends to be more peaked near zero; this gives the characteristic Laplace distribution shape.

4 Comparative Results

Our experiments involved compressing 28 eight-bit grayscale images. We report our results both in terms of compression ratio (original size divided by compressed size) and in terms of a new measure called *compression gain*, described in Section 4.1. In Section 4.2 we briefly describe the test images and the compression methods compared, and we present the results in narrative, tabular and graphical form.

4.1 Compression gain

A robust measure of compression performance, called *compression gain*, is introduced in [2]. Here we give a slightly different formulation, using a more convenient scale. We define the compression gain by

$$\text{compression gain} = 100 \log_e \frac{\text{reference size}}{\text{compressed size}}, \tag{2}$$

where the reference size and compressed size are expressed in the same units, usually either total bytes in the file or average bits per pixel. Compression gain is expressed in *percent log ratio*, denoted by the $\frac{\circ}{\circ}$ sign. Since $\log_e(1+x) \sim x$ for small x, a difference of any given small percent log ratio means almost the same thing as the same difference expressed as an ordinary percentage. For the reference size we may choose the original file size, the zero-order entropy of the file, or the size of the compressed file produced by some standard lossless compression method.

The logarithm in Equation (2) gives us additivity, so we can simply subtract the compression gains of two methods to find the difference between them, expressed in a percentage-like quantity. Coding effects, often expressed as percentages of code length, can be given as losses that can simply be subtracted from the compression gain of the modeling method. In this form we can clearly separate coding and modeling, and we can see how tiny the coding effects are. The *gain/loss* terminology is natural: larger numbers mean better compression.

4.2 Test procedures and results

The 28 images comprising our test data include 21 Landsat Thematic Mapper images and seven other images (the "USC images") widely used in compression studies. There are three Landsat data sets, each consisting of seven images (spectral bands); the locations are Washington, D.C., Donaldsonville, Louisiana (90 kilometers west of New Orleans), and Ridgely, Maryland (70 kilometers southeast of Baltimore). Each band is coded independently of the other bands; we do not attempt to make use of the correlation between bands in a data set. All images consist of 512×512 8-bit grayscale pixels except the Ridgely images, which are 368×468 pixels. Each of the Landsat data sets contains one highly compressible image (compressible to better than $8:1$) and six "normal" images.

In Table 1 we report compression ratios for all 28 images using a number of compression methods, including three versions of the MLP algorithm. The same results

are presented as compression gains in Table 2, and in graphical form in Figure 3. To avoid cluttering the graphs in Figure 3 we include results for only one version of MLP, and to keep the scale large we omit data for the three highly compressible Landsat images.

The CCITT/ISO Joint Photographic Experts Group (JPEG) has recently proposed a standard for image compression [1] that addresses both lossless and lossy compression. We compare our results with the lossless mode of the JPEG proposed standard. The JPEG lossless mode is based on prediction by one, two, or three points, followed by Huffman coding or arithmetic coding; encoding proceeds in raster scan order. The standard gives some latitude to implementors; we report results from an implementation based on two-point prediction with arithmetic coding. For all except the three highly compressible Landsat images, two-point prediction provides $4\frac{0}{0}$ to $10\frac{0}{0}$ more compression gain than three-point prediction. We also include results for the Minimax coder, AT&T's original lossless compression submission to the JPEG, and for the UNIX *compress* program. Figures for the Minimax coder are available only for the Landsat images, not the USC images.

We report results for three versions of the MLP algorithm. The first is the original MLP algorithm, with limited error modeling: we assume Laplace distributions and compute and transmit a single variance for each level. The second also uses Laplace distributions, and estimates variances using the variability index technique, with scaling factor $f = 0.992$. In the third we use Equation (1) with n starting at 1.5 in each level, falling to $n = 1$ (Laplace) by the end of each level. This is the method included in Figure 3. As noted in Section 2, there is only a small improvement when we optimize n for each image or vary the exponent within each level. Optimizing for each image requires compressing each image many times, so we omit that step in our reported results; varying the exponent within each level is a simple procedure that does improve compression slightly, so we include that step.

Here we summarize the methods tested together with the abbreviations used in Tables 1 and 2 and Figure 3.

Abbreviation	Compression Method
JPEG	JPEG lossless mode, with two-point prediction
Original	Our MLP method, using Laplace distributions and a single variance for each level
$n = 1$	Our MLP method, using Laplace distributions and variance estimation by variability index
$n = 1.5$ to 1 (MLP–VI)	Our MLP method using the variability index, with the exponent varying from 1.5 to 1.0 in each level
MM coder	AT&T Minimax coder, with two-point prediction
compress	UNIX *compress* program

In our experiments we find that using MLP with the variability index technique gives the best compression for all images; it is about 7 percent better than JPEG

Image	JPEG	Multi-level Progressive (MLP) Original	MLP Variability Index $n = 1$	MLP Variability Index $n = 1.5$ to 1	MM	*compress*
W1	2.07	2.16	2.20	2.21	2.15	1.70
W2	2.67	2.74	2.82	2.83	2.76	2.21
W3	2.28	2.37	2.43	2.44	2.36	1.92
W4	1.81	1.90	1.92	1.93	1.88	1.46
W5	1.68	1.76	1.78	1.80	1.74	1.34
W6	7.92	8.66	9.65	9.74	8.08	5.36
W7	2.10	2.20	2.22	2.23	2.17	1.70
D1	2.26	2.27	2.40	2.41	2.34	1.79
D2	3.07	2.92	3.23	3.24	3.16	2.36
D3	2.58	2.45	2.72	2.72	2.65	1.99
D4	1.85	1.89	1.97	1.98	1.91	1.34
D5	1.82	1.83	1.94	1.95	1.87	1.34
D6	9.25	9.61	11.14	11.25	9.35	6.14
D7	2.17	2.21	2.30	2.31	2.23	1.65
R1	2.28	2.40	2.44	2.46	2.38	1.79
R2	2.94	3.04	3.14	3.17	3.07	2.26
R3	2.45	2.55	2.64	2.66	2.56	1.86
R4	2.24	2.32	2.41	2.42	2.33	1.76
R5	1.78	1.85	1.93	1.94	1.85	1.34
R6	2.08	2.17	2.23	2.25	2.15	1.58
R7	7.43	8.06	8.70	8.76	7.85	5.50
Landsat average	2.41	2.48	2.58	2.59	2.50	1.87
couple	1.54	1.57	1.64	1.64		1.17
crowd	1.87	1.79	2.03	2.03		1.31
lax	1.31	1.34	1.38	1.38		1.04
lena	1.72	1.80	1.89	1.89		1.14
man	1.64	1.64	1.75	1.75		1.15
woman1	1.58	1.58	1.67	1.67		1.30
woman2	2.28	2.38	2.51	2.50		1.40
USC average	1.66	1.68	1.78	1.78		1.20

Table 1: Compression ratios (original file size divided by compressed file size) for 28 grayscale images. Figures for the Minimax coder are not available for the USC images.

Image	JPEG	Multi-level Progressive (MLP) Variability Index			MM	*compress*
		Original	$n = 1$	$n = 1.5$ to 1		
W1	—	4.0	5.9	6.3	3.6	−19.9
W2	—	2.4	5.4	5.8	3.4	−18.9
W3	—	3.6	6.2	6.6	3.5	−17.2
W4	—	4.6	6.0	6.7	4.0	−21.6
W5	—	4.9	6.1	6.7	3.6	−22.6
W6	—	8.9	19.8	20.7	2.0	−39.1
W7	—	4.6	5.8	6.3	3.5	−21.2
D1	—	0.2	6.0	6.4	3.2	−23.5
D2	—	−5.0	5.0	5.2	2.8	−26.6
D3	—	−5.3	5.2	5.4	2.7	−26.1
D4	—	2.4	6.5	7.1	3.2	−32.1
D5	—	0.5	6.4	6.8	2.8	−30.6
D6	—	3.9	18.6	19.6	1.1	−41.0
D7	—	1.8	5.9	6.5	2.7	−27.3
R1	—	5.1	6.7	7.6	4.5	−24.3
R2	—	3.6	6.9	7.8	4.4	−26.0
R3	—	3.9	7.2	8.1	4.1	−27.7
R4	—	3.7	7.4	8.0	4.1	−23.8
R5	—	3.7	7.6	8.3	3.7	−28.3
R6	—	4.4	7.1	8.0	3.7	−27.3
R7	—	8.2	15.8	16.5	5.5	−30.1
Landsat average	—	2.6	6.7	7.3	3.4	−25.3
couple	—	2.0	5.9	6.3		−27.4
crowd	—	−4.3	8.4	8.3		−35.4
lax	—	2.0	5.1	5.3		−23.5
lena	—	4.3	9.3	9.4		−41.8
man	—	0.0	6.7	6.7		−35.2
woman1	—	−0.1	5.6	5.5		−19.2
woman2	—	4.2	9.3	9.2		−48.8
USC average	—	1.1	7.0	7.0		−32.3

Table 2: Compression gains for 28 grayscale images, expressed in percent log ratio ($\frac{\%}{\%}$) using JPEG lossless mode compression as the reference value. Figures for the Minimax coder are not available for the USC images.

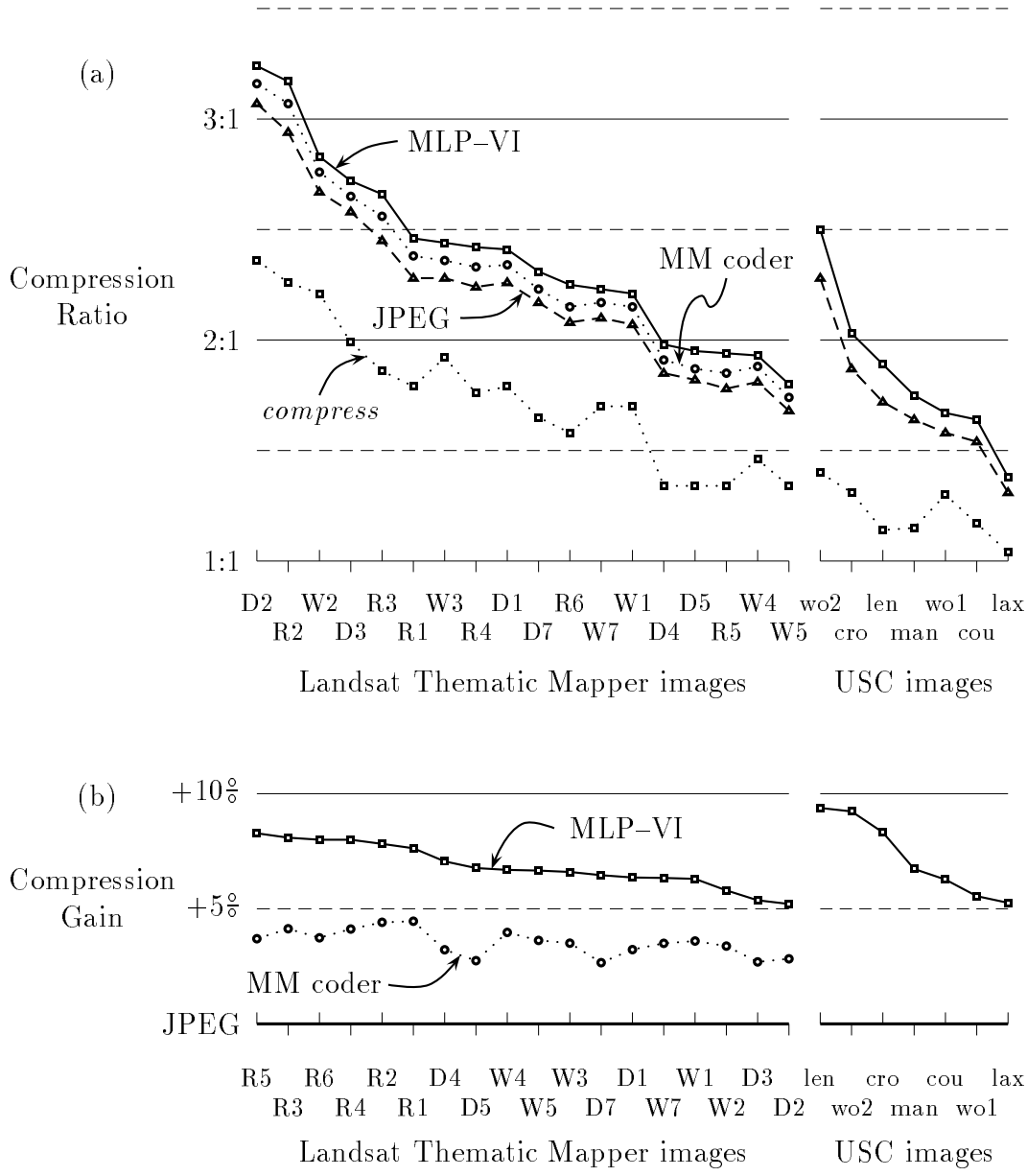

Figure 3: (a) Compression ratios (original size divided by compressed size) arranged in order of decreasing compressibility, measured by the compression ratio using MLP–VI. (b) Compression gains measured with respect to JPEG lossless mode, expressed in percent log ratio ($\frac{\circ}{\circ}$) and arranged in order of decreasing compressibility as measured by the compression gain using MLP–VI. Landsat images are identified by data set and band number. Images D6, W6, and R7 are not included: they are all highly compressible, and would appear off the scale in both graphs. The USC images are identified by the first three letters (or first two letters and last digit) of their names. Figures for the Minimax coder are not available for the USC images.

lossless mode and almost 4 percent better than the Minimax coder. The variability index technique improves on the original MLP algorithm by over 4 percent, and using a different exponent in Equation (1) gives another half percent compression, at least for the Landsat images. Figures for the *compress* command are included only to show how poorly suited it is for image compression.

As we have noted, the MLP method runs slowly. The prediction step, the computation of the variability index, and arithmetic coding all make heavy use of the processor. However, hardware continues to become faster and cheaper. A typical run of the MLP program takes about 2 minutes and 26 seconds on a Sun SPARCstation 1, the equipment with which we began this research; now we can run it in about 43 seconds on a SPARCstation 10, an improvement factor of 3.4. It really does make sense to try for maximum compression; the hardware will catch up!

5 Conclusions

We conclude that using the variability index for the error modeling component of the MLP algorithm is extremely effective. It gives good estimates of local pixel variances without requiring any overhead (side information) to describe the models. In addition, the use of the variability index to classify pixels according to their local variability allows us to see that prediction errors are best modeled by a Laplace distribution only for groups of pixels with low variability; this leads us in turn to the use of different distributions. Since the original MLP algorithm gives results generally better than the JPEG lossless mode and comparable to the AT&T Minimax coder, and the variability index technique gives more than 4 percent better compression than the original method, we see that the variability index technique improves substantially on other available lossless image compression methods. It also retains MLP's advantages of progressivity and parallelizability.

We are currently investigating several questions. We wish to find a theoretical basis for the variability index; we expect that this will enable us to estimate the variance directly from the variability index, without the sorting step and the indirect adaptive estimation. We are also investigating the choice of exponent in Equation (1); an understanding of the factors that combine to produce the optimal exponent at a given pixel will enable us to estimate the exponent directly, leading to still better compression. A related issue is to determine the distribution mixing process that leads to the observed error distributions.

In other recent work [3,5], we identify simplifications that enable us to obtain maximum speed at some cost in compression efficiency. The resulting non-progressive compression system (called *FELICS*) gives compression about $7\frac{0}{0}$ worse than MLP and about the same as the JPEG lossless mode. FELICS runs nearly 50 times faster than MLP, five times faster than a commonly used implementation of the JPEG lossless mode, and as fast as the UNIX *compress* program. We have also investigated a progressive version of FELICS [4] that uses MLP's pixel sequence; it obtains about $1\frac{0}{0}$ better compression than non-progressive FELICS at the cost of a small decrease in speed.

Acknowledgments. We wish to thank James C. Tilton of the NASA Goddard Space Flight Center for supplying the images used for our experiments. We also wish to thank Allan R. Moser of E. I. duPont de Nemours and Company (Inc.) for assisting us with experimental evaluation of lossless mode JPEG compression.

References

[1] "Digital Compression and Coding of Continuous-Tone Still Images, Part 1, Requirements and Guidelines," ISO/IEC JTC1 Draft International Standard 10918–1, Feb. 1991.

[2] P. G. Howard & J. S. Vitter, "New Methods for Lossless Image Compression Using Arithmetic Coding," *Information Processing and Management* 28 (1992), 765–779.

[3] P. G. Howard & J. S. Vitter, "Design and Analysis of Fast Text Compression Based on Quasi-Arithmetic Coding," *Information Processing and Management* 30 (1994), 777–794, also appears in shorter form in the proceedings of the Data Compression Conference, J. A. Storer and M. Cohn, eds., Snowbird, Utah, March 30-April 1, 1993, 98-107.

[4] P. G. Howard & J. S. Vitter, "Fast Progressive Lossless Image Compression," in *Image and Video Compression Conference, Symposium on Electronic Imaging: Science & Technology* #SPIE–2186, San Jose, California, Feb. 9-10, 1994, 98–109.

[5] P. G. Howard & J. S. Vitter, "Arithmetic Coding for Data Compression," *Proc. IEEE* 82 (June 1994), 857–865.

[6] J. B. O'Neal, "Predictive Quantizing Differential Pulse Code Modulation for the Transmission of Television Signals," *Bell Syst. Tech. J.* 45 (May–June 1966), 689–721.