JPEG Quality Transcoding using Neural Networks Trained with a Perceptual Error Measure

John Lazzaro and John Wawrzynek
CS Division, UC Berkeley
Berkeley, CA 94720-1776
lazzaro@cs.berkeley.edu, johnw@cs.berkeley.edu

Abstract

A JPEG Quality Transcoder (JQT) converts a JPEG image file that was encoded with low image quality to a larger JPEG image file with reduced visual artifacts, without access to the original uncompressed image. In this paper, we describe technology for JQT design that takes a pattern recognition approach to the problem, using a database of images to train statistical models of the artifacts introduced through JPEG compression. In the training procedure for these models, we use a model of human visual perception as an error measure. Our current prototype system removes 32.2% of the artifacts introduced by moderate compression, as measured on an independent test image database using a perceptual metric; this improvement results in an average PSNR reduction of 0.634 dB.

1. INTRODUCTION

JPEG is a lossy compression algorithm for digital images (Wallace, 1992). An image file format that uses JPEG compression, JFIF, has become the standard image file format for the World Wide Web and for digital cameras. The JPEG encoding algorithm gives users direct control over the compression process, supporting tradeoffs between image quality and degree of compression. Higher compression ratios may result in undesirable visual artifacts in the decoded image.

Given a JPEG-encoded image that was compressed to a small size at the expense of visual quality, how can we reduce visual artifacts in the decoded image? A substantial body of literature addresses this question (Wu and Gersho, 1992; Jarske et al, 1994; Ahumada and Horng, 1994; Minami and Zakhor, 1995; Yang et al, 1995; O'Rourke and Stevenson, 1995). In these references, artifact reduction is undertaken as part of a JPEG decoder.

In this paper, we consider image artifact reduction as part of a different application: a JPEG Quality Transcoder (JQT). A JQT converts a JPEG image file that was encoded with low image quality to a larger JPEG image file with reduced visual artifacts, without access to the original uncompressed image. A JQT should perform only the lossless part of the JPEG decoding algorithm, followed by signal processing on the partially decompressed representation, followed by lossless JPEG encoding to produce the transcoded image. A JQT provides a simple way to improve image quality in situations where modifying the JPEG encoding or decoding operations is not possible. Applications of a JQT include enhancing the quality of JPEG images accessed from an Internet proxy server, reducing artifacts of video streamed from a motion-JPEG server, and improving the "number of stored photos" vs. "image quality" tradeoff of digital cameras.

In contrast to most previous work in artifact reduction, we take a pattern recognition approach, using a database of images to train statistical models of artifacts. In the training procedure for these models, we use a model of human visual perception as an error measure.

The paper is organized as follows. In Section 2, we review the JPEG compression system. Section 3 introduces the general architecture of the JQT. Section 4 describes the human visual system error measure. Section 5 explains the detailed architecture of our statistical artifact models. Section 6 details the training of the models. Section 7 and 8 shows data from a JQT using these models. Section 9 offers suggestions for further research.

2. JPEG AND JFIF

This section reviews JPEG compression and the JFIF file format (Wallace, 1992). Cathode ray tube (CRT) color computer display hardware has a natural color representation, RGB, consisting of 3 numbers that code the linear excitation intensity of red (R), green (G), and blue (B) phosphors at each pixel. High-quality display hardware uses an 8 bit value to encode each color plane (R, G, and B) for a 24-bit pixel encoding. Note that in practice, a nonlinear RGB encoding is often used with JPEG to improve performance. In this paper, however, we use linear RGB image coding throughout.

The JPEG algorithm compresses each color plane of an image independently. Since cross-plane correlations cannot be captured in such a scheme, color representations with low correlation between planes are a good match to JPEG. The RGB coding has relatively high correlation between planes. A color scheme with lower cross-plane correlation, YC_bC_r, is the color code for the JFIF file format. The YC_bC_r code includes the luminance plane Y, which codes a monochrome version of the image in 8 bits. The 8-bit C_b and C_r planes code chrominance information. A linear transformation converts between RGB and YC_bC_r coding.

In addition to lower cross-plane correlation, the YC_bC_r color code has another advantage for image compression. The human visual system is less sensitive to high spatial frequency energy in the chrominance planes of an image, relative to the luminance plane. To exploit this phenomenon, the JFIF file encoding process begins by subsampling the C_b and C_r planes of a YC_bC_r image by a factor of two in both horizontal and vertical dimensions. This subsampling yields an immediate

compression of nearly 60% with little degradation in image quality.

After color transformation to YC_bC_r and chrominance subsampling, JFIF file encoding continues by applying the JPEG encoding algorithm to each plane separately. This encoding begins by dividing the image plane into a grid of non-overlapping blocks of 8 by 8 pixels; each block is coded independently. Encoding begins by taking the two dimensional Discrete Cosine Transform (DCT) on each pixel block P, yielding an 8 by 8 block of coefficients K, defined as

$$k(u,v) = \frac{C(u)C(v)}{4} \sum_{x=0}^{7} \sum_{y=0}^{7} p(x,y) \cos((2x+1)u(\pi/16)) \cos(2y+1)v(\pi/16)), \quad (1)$$

where $u = 0, \ldots, 7$ and $v = 0, \ldots, 7$. The term $C(i) = 1/\sqrt{2}$ if $i = 0$, $C(i) = 1$ elsewise. In this equation, $p(x,y)$ is the value at position (x,y) in the pixel block P, and $k(u,v)$ is the value for frequency (u,v) in the coefficient block K. Coefficient $k(0,0)$ codes the DC energy in the block; other coefficients $k(u,v)$ are AC coefficients, coding spatial frequency energy. 11-bit $k(u,v)$ values are needed to accurately code 8-bit $p(x,y)$ values.

Most of the energy in real-world images lies in the lower spatial frequency coefficients. In addition, the sensitivity limits of the human visual system vary with spatial frequency. Careful quantization of $k(u,v)$ values can exploit these two phenomena, yielding a considerable reduction in the bitsize of a coefficient block while maintaining good image quality. Coefficient quantization is the sole lossy step in the JPEG encoding algorithm. Each coefficient $k(u,v)$ is divided by the quantization divisor $q(u,v)$; the dividend is rounded to the nearest integer, yielding scaled quantized coefficients. In baseline JPEG encoding, each plane of an image uses a single matrix Q of $q(u,v)$ values to quantize all blocks in the plane.

The JPEG encoding process concludes by lossless compression of the scaled quantized coefficients, yielding a bit-packed JFIF file that contains coefficient information for each block of each plane, and the quantization matrix for each plane.

JFIF file decoding begins with lossless decompression of the coefficient blocks and quantization matrices for each plane. For each coefficient block, each scaled quantized coefficient is multiplied by the appropriate quantization divisor $q(u,v)$, producing the quantized coefficient $\hat{k}(u,v)$. The pixel block is then reconstructed from the quantized coefficient block, via the Inverse DCT:

$$\hat{p}(x,y) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)C(v)\hat{k}(u,v) \cos((2x+1)u(\pi/16)) \cos(2y+1)v(\pi/16)). \quad (2)$$

In this way, a complete image for each color plane is reconstructed block by block. Replication of the subsampled C_b and C_r planes, and conversion from YC_bC_r to RGB, complete the decoding process.

3. A JPEG QUALITY TRANSCODER (JQT)

Using the definitions of the last section, we now review artifact reduction algorithms for JPEG compression. If the reconstructed image is perceptually different from the original image, visual artifacts have been introduced during coefficient quantization. Published methods for artifact reduction as part of the JPEG decoding process use a combination of these methods to improve image quality:

• Linear or nonlinear image processing on the reconstructed image, to lessen the visual impact of artifacts (Minami and Zakhor, 1995; Jarske et al, 1994).

• Replacing the decoding algorithm as described in Equation 2, with an iterative (Yang et al, 1995; O'Rourke and Stevenson, 1995) or codebook (Wu and Gersho, 1992) approach.

• Pre-processing the quantized coefficients before proceeding to JPEG decoding as defined in Equation 2 (Ahumada and Horng, 1994; Minami and Zakhor, 1995).

The final method is the preferred approach for implementing artifact reduction in a JPEG Quality Transcoder; the first two methods would require the large overhead of decoding and re-encoding the image.

Previous work in pre-processing quantized coefficients for artifact reduction (Minami and Zakhor, 1995; Ahumada and Horng, 1994) uses the tools of iterative optimization. In these papers, metrics are developed that measure the severity of a class of JPEG artifacts. These metrics are then used in an iterative optimization algorithm to calculate coefficient values that minimize image artifacts.

In this paper, we pursue a different approach for pre-processing quantized coefficients for artifact reduction. The approach rests on the assumption that the information lost during quantization of coefficient $k(u,v)$ in color plane C of block (i,j) of an image, expressed as $k_{ij}^C(u,v) - \hat{k}_{ij}^C(u,v)$, can be accurately estimated from other information in the compressed image.

We use multi-layer perceptrons to estimate $k_{ij}^C(u,v) - \hat{k}_{ij}^C(u,v)$. These networks are convolutional in input structure: the same network is used for each (i,j) block in an image, and inputs to the network are selected from the coefficient blocks of all three color planes in the neighborhood of block (i,j). Quantization divisor matrices are also used in the estimation process.

We use a total of 64 neural networks, each specialized in architecture (number of hidden units, selection of inputs, etc.) for a particular spatial frequency (u,v). Each network has three outputs, one for each color plane. We detail the network architecture and training procedure in Sections 5 and 6. A key part of the training procedure is the computation of the error, as perceived by a human observer, between corresponding color pixels in an original image and a reconstructed image. In the next section of the paper, we review the literature of perceptual visual measures, and present the perceptual error measure.

4. A PERCEPTUAL ERROR METRIC

In this section, we describe a pointwise perceptual metric, computed on a pixel (Y, C_b, C_r) in an original image and the corresponding pixel $(\hat{Y}, \hat{C}_b, \hat{C}_r)$ in a reconstructed image. In Appendix 2, we present the exact formulation of the metric.

Our goal is to develop a metric that is a good predictor for human sensitivity to the types of color imaging errors introduced in JPEG encoding. A recent paper (Fuhrmann et al, 1995) also addresses this issue, in the context of monochrome imaging (Y plane only). The (Fuhrmann et al, 1995) paper describes a set of psychophysical experiments that measures the threshold and suprathreshold sensitivity of subjects to JPEG-induced errors. The data from these experiments are compared with the predictions of a large collection of image metrics. While mean-squared er-

ror (defined as $|Y-\hat{Y}|^2$) is shown not to be a good predictor of human performance, distortion contrast (defined as $|Y-\hat{Y}|/(Y+\hat{Y}+C)$) is highly predictive.

We cannot use distortion contrast directly as our training metric, as our task involves the measurement of error in *color* images. A good extension of monochrome contrast that has a firm basis in color science is the cone contrast metric (Cole et al, 1993). This metric is computed in the LMS color coordinate space. As the RGB color space is the coordinate system derived from the spectral sensitivity functions of CRT screen phosphors, the LMS color space is the coordinate system derived from the spectral sensitivity of photopigments of the long-wavelength sensitive (L), medium-wavelength sensitive (M) and short-wavelength sensitive (S) cones in the human retina.

A simple linear transformation, shown in Appendix 1, converts (Y, C_b, C_r) and $(\hat{Y}, \hat{C}_b, \hat{C}_r)$ pixel values to (L, M, S) and $(\hat{L}, \hat{M}, \hat{S})$. To compute the cone contrast vector $(\Delta L/L, \Delta M/M, \Delta S/S)$ from the original pixel and reconstructed pixel LMS values, we use the equations:

$$\Delta L/L = \frac{L - \hat{L}}{L + L_o}$$

$$\Delta M/M = \frac{M - \hat{M}}{M + M_o}$$

$$\Delta S/S = \frac{S - \hat{S}}{S + S_o}.$$

The constants L_o, M_o, and C_o model a limitation of CRT displays: a pixel position that is programmed to produce the color black actually emits a dim grey color (Macintyre and Cowan, 1992). The constants L_o, M_o, and C_o represent this grey in LMS space.

Cone contrast space has an interesting psychophysical property (Cole et al, 1993) revealed by the experiment of briefly flashing a slightly off-white color $(\hat{L}, \hat{M}, \hat{S})$ on the white background (L, M, S) and measuring the detection threshold of the off-white color, for many different off-white shades. The detection threshold can be shown to be the result of three independent mechanisms, and each mechanism can be expressed as a linear weighting of the cone contrast representation $(\Delta L/L, \Delta M/M, \Delta S/S)$. These mechanisms correspond to the familiar opponent channels of Red-Green (RG), Blue-Yellow (BY), and Black-White (BW). In our metric, we compute these three opponent channel values from the cone contrast vector. The BW channel is qualitatively similar to the distortion contrast metric in (Fuhrmann et al, 1995). The other two channels (RG and BY) code chrominance information in a contrast framework.

The opponent coding is a suitable representation for incorporating the effects of visual masking into our metric. Visual masking is the phenomena of errors being less noticeable around an image edge, and more noticeable around the smooth parts of an image. In our metric, we only model masking in the luminance plane. We weight the BW output by an activity function $A(x, y)$, that is unity for pixel positions in the smooth regions of original image and less than unity for pixel positions near an edge (Kim et al, 1996). The activity function can be computed once for each

pixel of each image in the database, and reused to calculate the error of different reconstructions of a pixels.

To complete our metric, we sum the weighted absolute values of the opponent channel outputs, yielding the final error function $E(Y, C_b, C_r; \hat{Y}, \hat{C}_b, \hat{C}_r)$. The weights correspond to the relative detection sensitivities of the underlying mechanisms, as measured in the (Cole et al, 1993) study. Our use of the absolute values of the opponent channel outputs, rather than the square of these outputs, reflects the assumed independence of these mechanisms.

The metric presented in (van der Branden Lambrecht and Farrell, 1996) shares several details with our work, including opponent channels and a masking model. Major differences include our use of cone contrast space to compute the opponent channels and our formulation of the model to be efficient in a neural-network training loop. A more detailed human visual system model has been successfully applied to JPEG image coding in (Westen et al., 1996).

5. NETWORK ARCHITECTURE

We use the perceptual metric described in the last section to train statistical models of the information lost during JPEG encoding. In this section, we describe these models in detail.

Our system has 64 neural networks, each dedicated to modeling the information loss for a coefficient frequency (u, v). Figure 1 shows a typical network. The network has three outputs, $O^Y(u, v), O^{C_r}(u, v), O^{C_b}(u, v)$, that predict a normalized estimate of $k^C_{ij}(u, v) - \hat{k}^C_{ij}(u, v)$ for the three color planes for block position (i,j). These output neurons, as well as all hidden units, use the hyperbolic tangent function as the sigmoidal nonlinearity (output range, -1 to +1). In our current implementation, network weights are stored as floating-point values, and network outputs are computed using floating-point math.

The network outputs $O^Y(u, v), O^{C_r}(u, v), O^{C_b}(u, v)$ predict the information lost during JPEG encoding. We use these outputs to compute coefficient values $\tilde{k}^C(u, v)$ with reduced artifacts, using the equation:

$$\tilde{k}^C(u, v) = \hat{k}^C(u, v) + 0.5q^C(u, v)O^C. \tag{3}$$

This approach ensures that only plausible predictions are made by the system. Recall that during JPEG encoding, each DCT coefficient $k(u, v)$ is divided by the quantization divisor $q(u, v)$ and rounded to the nearest integer. During decoding, integer multiplication by $q(u, v)$ produces the quantized coefficient $\hat{k}(u, v)$. Note that this $\hat{k}(u, v)$ could have been produced by $k(u, v)$ values in the range of $\hat{k}(u, v) \pm 0.5q(u, v)$. Equation 3 only produces $\tilde{k}^C(u, v)$ values in this range.

In this paper, we model the information loss produced by a single set of quantization divisors: the quantization divisor tables recommended in Section K.1 of CCITT Rec T.81 Standards Document that defines JPEG. This restriction simplifies the artifact reduction system by eliminating the need to include quantization divisor inputs into the neural networks. These quantization divisors, which we refer to as Q_s in this document, produce good compression ratios with moderate visual artifacts, and have been adopted by many popular applications.

The output neurons in Figure 1 receive inputs from a pool of hidden-layer units. Each hidden-layer unit receives a set of coefficient inputs, selected from the coefficient blocks of one color plane in the neighborhood of block (i,j). We replicate the chrominance coefficient blocks to match the sampling pitch of the luminance block, to simplify the network input architecture. Each coefficient input is divided by its variance, as computed on the training set. Each hidden-layer and output unit has a bias input, not shown in Figure 1.

In Figure 1, the chosen inputs are drawn in black on the coefficient block grids. The receptive fields are drawn to be correct for coefficient 8 ($u = 1, v = 2$) as labeled in Figure 2b – note that the receptive fields all include this coefficient. We hand-crafted these receptive fields, guided by pilot training experiments.

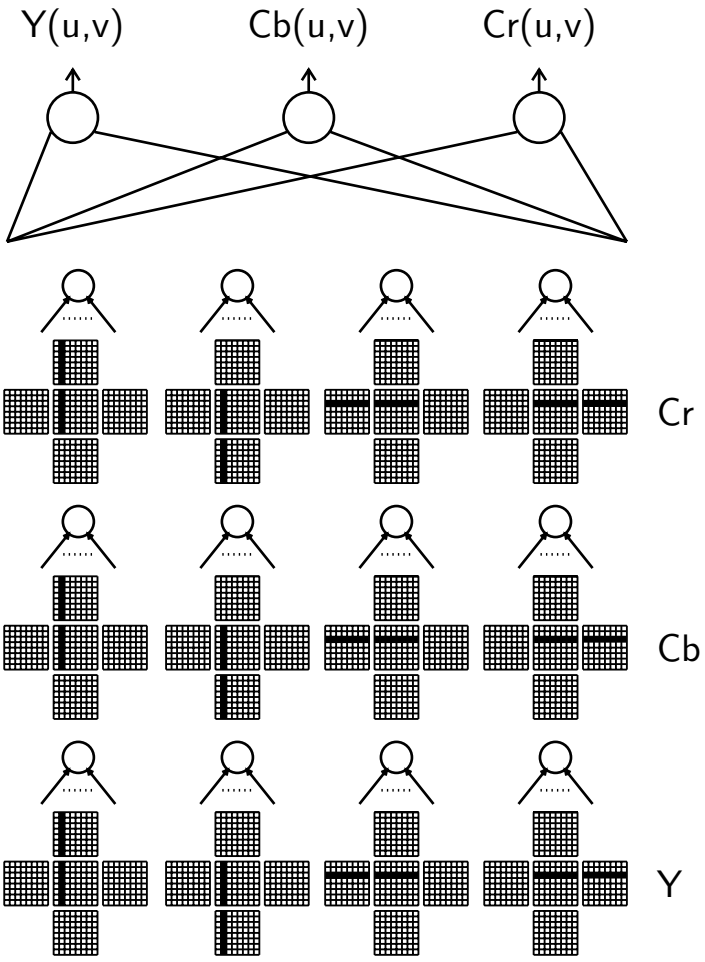
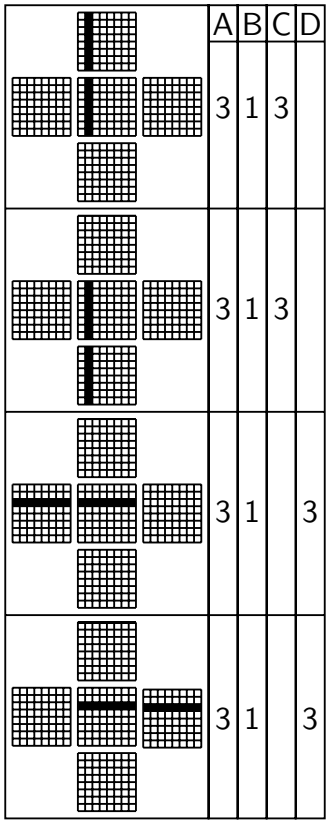


Figure 1. A typical neural network for modeling information loss of coefficient (u, v). Outputs marked $C(u, v)$ (and notated $O^C(u, v)$ in the main text) predict a normalized estimate of $k^C_{ij}(u, v) - \hat{k}^C_{ij}(u, v)$, and receive input from all 12 hidden units. Hidden units specialize on a block edge for a single color plane. See caption for Figure 2 for explanation of notation used to graphically denote hidden unit receptive fields. The network as drawn corresponds to architecture B in Figure 2.

The following ideas underly the good performance of these receptive fields:

• A brute-force receptive field pattern would simply include all 64 coefficients for the center coefficient block and the four neighboring blocks, for a total of 320 coefficients. Experiments using these types of receptive fields yielded poor results: most inputs were irrelevant for constructing a useful feature for the task, and the presence of these useless inputs confused the learning process. It was necessary to pre-select a small subset of inputs from the universe of 320, that carried information for a certain class of features.

• A natural way to divide hidden-unit space is to let hidden units specialize in artifacts occurring on one edge of the center coefficient block. These hidden units would only receive inputs from the center block and one adjacent block, paring the original universe of 320 potential inputs down to 128 inputs. Note that all the receptive fields shown in Figure 1 have this characteristic.



	A	B	C	D
	3	1	3	
	3	1	3	
	3	1		3
	3	1		3

(a)

v \ u	0	1	2	3	4	5	6	7
0	0	1	5	6	14	15	27	28
1	2	4	7	13	16	26	29	42
2	3	8	12	17	25	30	41	43
3	9	11	18	24	31	40	44	53
4	10	19	23	32	39	45	52	54
5	20	22	33	38	46	51	55	60
6	21	34	37	47	50	56	59	61
7	35	36	48	49	57	58	62	63

(b)

Figure 2. (a) Table showing networks architectures (A – D) used in the system. Column entries note the number of hidden units of each receptive field type in the network. Receptive field drawings show the five adjacent coefficient blocks for a single color plane – $(i, j), (i \pm 1, j)$ and $(i, j \pm 1)$ – drawn as a cross, with the selected inputs drawn in black. The receptive fields are drawn for coefficient 8 ($u = 1, v = 2$) as labeled in Figure 2b. **(b)** The zig-zag coefficient numbering convention.

• Experiments using these specialized hidden-units suggested that hidden-units specializing in horizontal artifacts (i.e. the left and right edges of the block) can combine information over the full range of horizontal spatial frequency coefficients, but have difficulty combining information over vertical spatial frequencies. A bar-shaped receptive field exploits this observation. We found that for a horizontally-specialized hidden unit for coefficient (u, v) a horizontal bar centered on v produced the best results. The receptive fields in Figure 1 show this pattern.

We used four different variants of the general architecture shown in Figure 1 in our work. Two variants differ in the number of copies of each of the 12 hidden units. We found that lower-frequency coefficients needed 3 copies of each hidden unit to best model the visual artifacts; conversely, higher-frequency coefficients sometimes worked best with a single copy of each hidden unit. These variants correspond to A and B in Figure 2a.

The other two variants are used only for coefficients $(0, u \neq 0)$ and $(v \neq 0, 0)$. These coefficients only have energy in one spatial frequency axis (horizontal or vertical). For some of these coefficients, the presence of hidden units specialized for the opposite spatial frequency axis results in degraded performance. For use with these coefficients, we use neural networks with only hidden units that specialize in the preferred axis; three copies of each hidden unit are used in these networks. These variants correspond to C and D in Figure 2a.

6. NETWORK TRAINING

We use backpropagation (Rumelhart et al, 1986) to train the networks. We train each of the 64 neural networks independently. Intuitively, one would expect simultaneous training of all 64 networks to produce better performance. However, in pilot experiments with using simultaneous training, we were not able to achieve good results. In this section, we describe the independent training method we use, and offer reasons for why we believe it works well in this application.

To train the neural network for coefficient (u, v) we proceed on a per-block basis. We begin by JPEG encoding the three planes of a pixel block in an original image; subsampled pixel blocks are used to encode C_b and C_r planes. We then compute the neural network outputs $O^Y(u, v), O^{C_r}(u, v), O^{C_b}(u, v)$ for coefficient (u, v).

Next, we compute a reconstructed pixel $\tilde{p}(x, y)$ in this block, under the assumption that only coefficient (u, v) has been quantized. This reconstruction can be computed efficiently using the equation

$$\tilde{p}^C(x, y) = p^C(x, y) + W_{xyuv}(\hat{k}^C(u, v) - k^C(u, v) + 0.5q^C(u, v)O^C(u, v)), \qquad (4)$$

where W_{xyuv} is the appropriate DCT coefficient for the pixel (x,y) and the coefficient (u,v). Note that due to chrominance subsampling, the $\tilde{p}^{C_b}(x, y)$ and $\tilde{p}^{C_r}(x, y)$ values are on a coarser (x, y) grid than the $\tilde{p}^Y(x, y)$. Pixel replication of the chrominance planes is necessary to produce registered YC_rC_b pixel values for the perceptual error calculation.

We measure the perceptual error of this reconstructed pixel, and update the weights of the neural network for coefficient (u, v) based on the error value. We repeat this "reconstruct, measure, update" loop for each of the 64 pixels in the block, to

complete a training cycle for a block of an image. Note that we compute $p^C(x, y)$ as a floating-point number, and retain floating-point precision for the perceptual error calculation.

By training the 64 neural networks independently, we provide the system with a simple problem to solve: cancelling the effect of a single coefficient quantization, in isolation from other coefficient quantizations, and without the roundoff noise of a complete inverse DCT computation. In addition, this approach offers 64-way parallelism for neural network training, and allows the incremental improvement of the artifact reduction system by upgrading a few of the 64 neural networks without requiring retraining of the rest.

Our image database consists of 699 color images, with an average dimension of 451 by 438 pixels. We collected these images from Internet archives. These images include natural scenes, face close-ups, and computer graphics images, and have not undergone previous lossy compression or subsampling. We divide the database into 3 parts: a training set of 347 images, a cross-validation set of 176 images, and a final test set of 176 images. We built our training software on top of the public-domain PVRG JPEG codec.

To train one of the 64 neural networks, we used two different methods of choosing the quantization divisors for the training images. The first method ("constant divisor") uses fixed divisor tables. We use the Q_s divisor tables scaled by 1.5, to exaggerate the artifacts and simplify the learning task. Pilot experiments showed that networks trained using scaled tables worked better on the artifacts induced by unscaled Q_s compression than networks trained with the unscaled Q_s tables.

However, there is a large variance in the measured perceptual error over the training set database compressed with a fixed divisor table. We found that for many coefficients, training the neural network with images quantized using a fixed table resulted in suboptimal performance.

As a result, we developed a second training method ("constant error"), where different images in the training set use different quantization divisors. We determined the quantization divisors using a multi-step process. First, we measured the average perceptual error E_{av} over the entire training set using Q_s. Then, for each image i in the training set, we found the value of K_i ($0.5 \leq K_i \leq 1.5$, in steps of 0.02), so that compression using the divisor tables K_iQ_s resulted in a perceptual error E_i such that $0.9E_{av} \leq E_i \leq E_{av}$. We used the divisor tables K_iQ_s for training image i; if no K_i could be found which met the inequality, the image was not used during training. This process resulted in a training set devoid of images with unusually large or small perceptual error; for most coefficients, the absence of these outlier images during training improved performance on the cross-validation set.

To train one of the 64 neural networks using the constant divisor or the constant error method, we initialize the weights of the network to random values, and measure the average $E(Y, C_b, C_r; \hat{Y}, \hat{C}_b, \hat{C}_r)$ value for the cross-validation set (defined as $\bar{E}^{cv}(u, v)$). We then train the network on each block of each image in the training set, using the per-block procedure described above. We set the initial learning rate to 1.0 for constant error training (0.001 for constant divisor training), and measure $\bar{E}^{cv}(u, v)$ at the end of each training pass. If $\bar{E}^{cv}(u, v)$ increases from the previous pass, we undo the weight updates from that training pass, reduce the learning rate by a factor of $\sqrt{10}$, and continue training. We terminate training when the learn-

ing rate falls below 0.0001 (0.00001 for coefficient (0,0)), and measure the average $E(Y, C_b, C_r; \hat{Y}, \hat{C}_b, \hat{C}_r)$ per pixel for the test set (defined as $\bar{E}^{tst}(u, v)$).

In our current system, the neural network architecture for each coefficient is chosen as follows. For each coefficient (u, v), we train all applicable network variants for each coefficient (see Section 5 and Figures 1 and 2 for details) with both training methods. We measure the cross-validation error $\bar{E}^{cv}(u, v)$ for each trained network, and pick the network with the lowest error. No artifact reduction is applied to a coefficient if all network architectures have a higher cross-validation error than the baseline error for the coefficient (i.e. the measured error when quantized using Q_s). After choosing the 64 networks for the final system, we measure the test-set error while correcting all 64 coefficients (defined as $\bar{E}^{tst}$, note the lack of a (u, v) specifier). In this final test, the coefficients for each block are computed using Equation 3, and normal JPEG decoding (Equation 2) is used to compute the pixels of the reconstructed images.

We trained our networks using a workstation cluster and a 4–CPU multiprocessor as our compute engines, powered by 250 MHz and 300 MHz UltraSPARC II processors. Depending on the network architecture, coefficient number, and training method, it took 30-120 minutes of computing time on an unloaded processor to compute a single epoch of training, and 3-15 epochs to completely train a network. To train the baseline system, we used approximately 5500 hours of processor time.

7. RESULTS: PERCEPTUAL ERROR PERFORMANCE

In this section, we describe the performance of the artifact reduction system on test-set images compressed using the Q_s quantization tables. In the description of these tables in Section K.1 of CCITT Rec T.81 Standards Document, it is noted that images compressed with the quantization table $Q_s/2$ (which we define to be Q_e) result in images that are usually indistinguishable from the source image. In light of this observation, a reasonable benchmark for our artifact reduction system is its success in reducing the perceptual error of an image compressed with Q_s to the perceptual error of the same image compressed with Q_e.

Figure 3 shows the performance of the artifact reduction system against this benchmark. It shows a plot of the perceptual error on the test set as a function of coefficient frequency. To produce these plots, we measure the error for quantizing one coefficient, while leaving the other coefficients unquantized. The frequency axis on this plot is a zig-zag scan of the (u,v) frequency space, as shown in Figure 2b.

Figure 3 shows three measurements. The bottom thick curve (labeled Q_e) shows JPEG decoding without artifact reduction, using Q_e divisor tables. The top thick curve (labeled Q_s) shows JPEG decoding without artifact reduction, using the Q_s tables. The thin line in Figure 3 (labeled JQT) shows the test-set performance of the artifact reduction system, while processing images compressed with the Q_s quantization tables.

In Figure 4, we replot this JQT performance curve in percentage terms, relative to the Q_s and Q_e performance values, using the expression

$$\% \text{ reduced} = 100 \, \frac{\mathrm{E(Q_s) - E(JQT)}}{\mathrm{E(Q_s) - E(Q_e)}}. \tag{5}$$

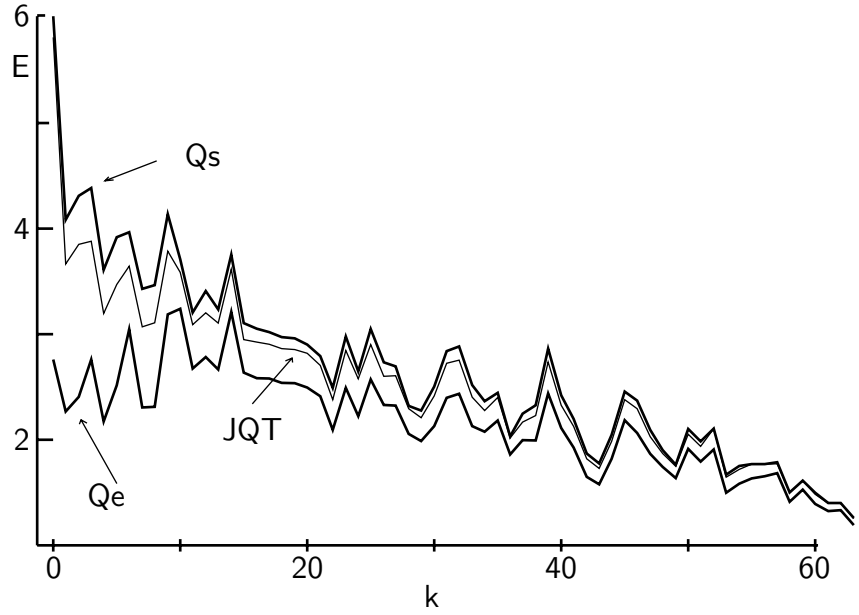


Figure 3. Plot showing $\bar{E}^{tst}(u,v)$ for JPEG encoding with Q_e quantizer tables (lower thick line), JPEG encoding with Q_s quantizer tables (upper thick line), and the results of the artifact reduction system when applied to the Q_s encoding (thin lines labeled JQT). Coefficient numbering scheme shown in Figure 2b.

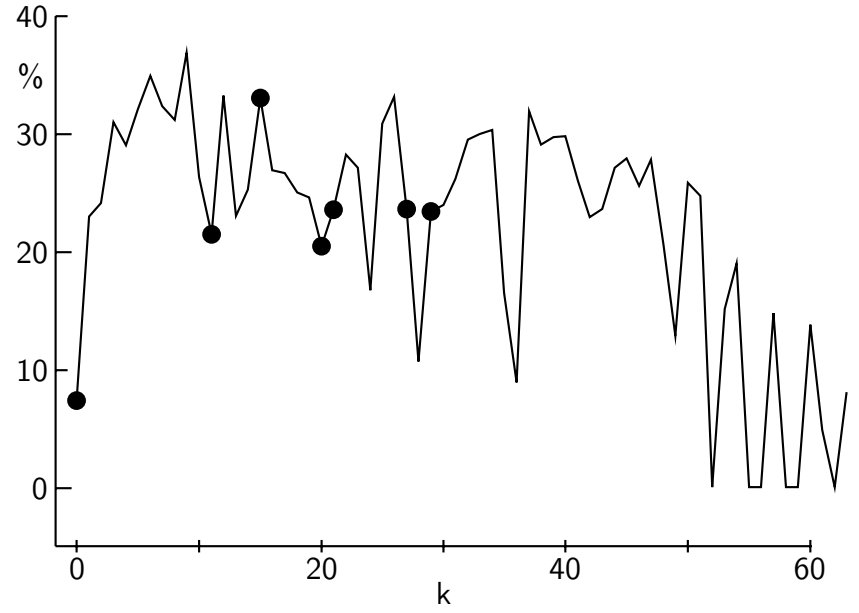


Figure 4. The percent reduction of perceptual error achieved by the artifact reduction system. Zero percent corresponds to Q_s error values, one hundred percent corresponds to Q_e error values. The dots indicate networks trained with the constant divisor training method; for all other coefficients, the cross-validation performance of a network trained with the constant error method was superior.

This plot shows 25% to 35% error reduction for most of the coefficients. Performance degrades for the lowest-frequency coefficients (due to the difficultly of the task) and for the highest-frequency coefficients (due to the limited amount of the training data, since natural images have very little energy at these spatial frequencies). The dip in performance for isolated midrange coefficient values corresponds to coefficients with the highest horizontal or vertical spatial frequency. This behavior can be seen more clearly in Figure 5(a), where we replot the percent reduction data on the two-dimensional u, v coefficient grid.

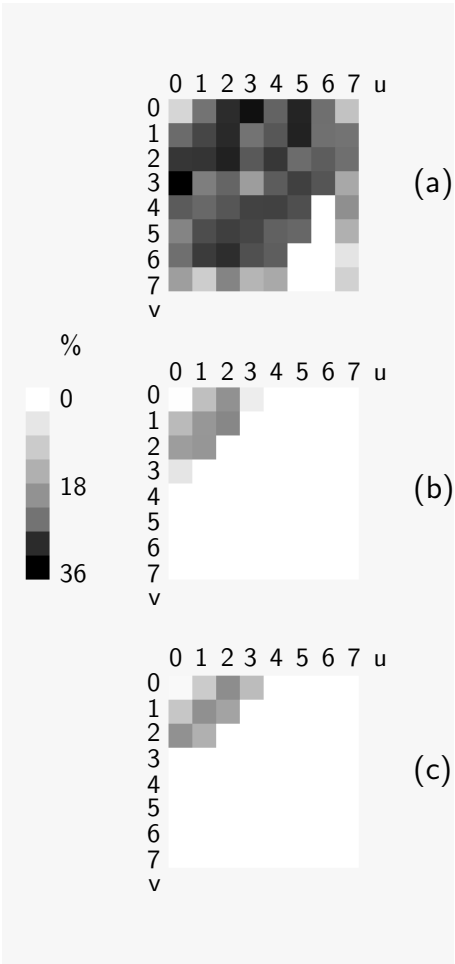


Figure 5. **(a)** The percent reduction for the artifact reduction system, plotted using gray scale on the (u, v) coordinate grid (see Figure 2b). Rule on side of Figure shows mapping of shading to percentage. **(b)** Percent reduction performance using a linear network instead of artifact reduction system; network trained with the perceptual error metric (See Section 8.4). Negative percentages mapped to white (0%). **(c)** Percent reduction performance using a linear network instead of artifact reduction system; network trained using mean square error (See Section 8.4). Negative percentages mapped to white (0%).

As described in Section 6, we used two different training methods (constant divisor and constant error) for each architecture; cross-validation performance was used to pick the final networks. For most coefficients, the constant error training method produced the best cross-validation performance; the dots shown in Figure 4 mark the exceptional coefficients whose highest-performing network was the product of constant divisor training. This result shows the advantage of excluding outlier images from the training set. We found that for architecture A, constant divisor training is particularly ineffective for higher-frequency coefficients; to save training time, we only trained architecture A using constant divisor training for coefficients 0–43.

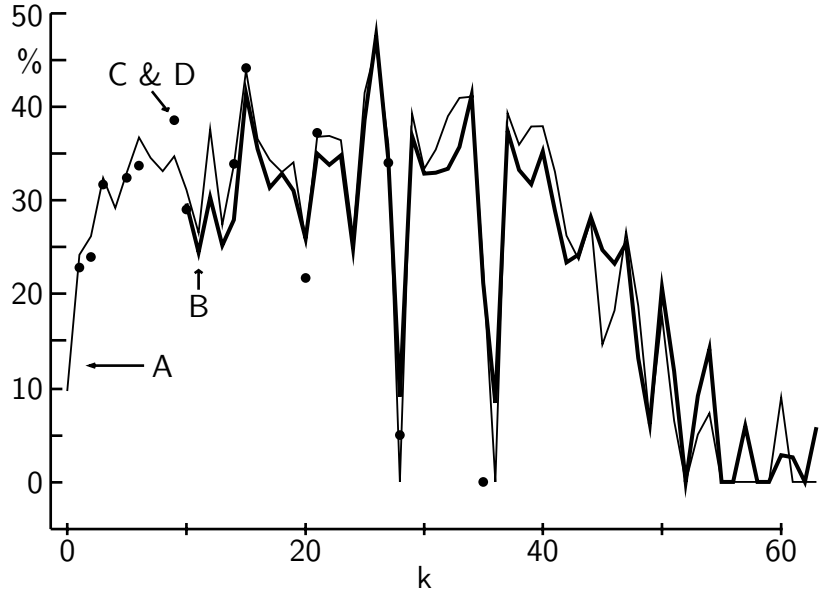


Figure 6. The percent reduction of perceptual error measured on the cross-validation, for architectures A (thin line), B (thick line), C and D (dots). For each coefficient number, the architecture with the highest percent reduction was chosen for the artifact reduction system.

v \ u	0	1	2	3	4	5	6	7
0	A	A	A	A	C	C	A	B
1	A	A	A	A	A	B	A	A
2	A	A	A	A	A	A	A	B
3	D	A	A	A	A	A	B	B
4	A	A	A	A	A	B		B
5	B	A	A	A	B	B		A
6	D	B	A	A	B			B
7	A	B	A	A	B			B

Figure 7. The network architecture (A–D) chosen for each coefficient, plotted on the (u, v) grid. A blank box indicates no network is used for this coefficient.

Figure 6 shows further data concerning the architecture selection process. In this graph, we show the performance on the cross-validation set for architectures A–D, for the training method with the higher performance. We show the cross-validation results because performance on this set is used to select the network architecture of the final system. Figure 7 shows the architecture selected for each coefficient.

The thin curve in Figure 6 shows the percentage error for architecture A (see Figure 2a), which has three copies of each hidden unit type. The thick curve shows the performance of architecture B, which has one copy of each hidden unit type. Architecture A works best for lower-frequency coefficients (below 25); to save training time, we did not train architecture B networks for the lowest-frequency coefficients (0–9). For higher frequency coefficients, the best architecture is coefficient dependent; as the graph shows, for some coefficients A is superior, for others B works better. This behavior is consistent with the theory that lower frequency artifacts are more complicated in nature, and are better modeled by higher-parameter models, whereas higher frequency coefficients have simpler artifact behavior, that may be overfit by a higher-parameter model.

The dots in Figure 6 show the cross-validation performance of architectures C and D. Recall that these architectures are used for the 14 coefficients ($(0, u \neq 0)$ and $(v \neq 0, 0)$) which have energy in only one spatial frequency axis. For the mid-frequency coefficients 9, 14, 15 and 21, these architectures perform better than architectures A and B on the cross-validation set: note the dots lie on or above the thin and thick lines at these locations in Figure 5. These networks generalized well, providing equivalent (14 and 15) or superior (9 and 21) performance to networks A and B on the test set.

Finally, we measure the perceptual error on the test set if all 64 coefficients are quantized simultaneously; this test simulates the performance of the artifact reduction system in a JPEG transcoder application. Without artifact reduction, this error is 0.02555 for Q_s quantization and 0.01965 for Q_e quantization. The test set error of the artifact reduction system while processing images compressed with the Q_s quantization tables is 0.02365 (defined as $\bar{E}^{tst}$ in Section 6). Equation 5 yields a percent reduction of 32.2% for this task. This result shows a good correlation between the improvements in single-coefficient performance shown in Figure 4.

8. RESULTS: STANDARD PERFORMANCE MEASURES

In Section 7 we report the performance of the artifact reduction system using the perceptual error metric. In this section, we characterize the system performance with techniques common in the image processing community.

8.1 PSNR

A classical way to judge image processing systems is the Peak Signal to Noise Ratio (PSNR). For a YC_rC_b image of size N by M, with original image pixels (Y, C_r, C_b) and degraded pixels $(\hat{Y}, \hat{C}_r, \hat{C}_b)$, the PSNR is defined as

$$10 \log_{10} \frac{255^2}{(1/3)(1/NM) \sum_{M,N} ((Y - \hat{Y})^2 + (C_r - \hat{C}_r)^2 + (C_b - \hat{C}_b)^2)}.$$

While this metric does not correlate well with human perception of artifacts (Fuhrmann et al, 1995), it is a common figure of merit in the image processing community. For each test-set image in the database used in Section 7, we measured the difference between the PSNR for compression using Q_s and compression using Q_e. The average PSNR difference between the two compression levels for an image in the test-set database is 2.5 dB. We also measured the difference between the PSNR for compression using Q_s and compression using Q_s followed by the artifact reduction system. The average PSNR difference is 0.63 dB, a significant portion of the 2.5 dB PSNR that corresponds to the perceptually-indistinguishable Q_e tables. Figure 8 tabulates this result, along with PSNR measurements for three color photos often used in the image processing community: Lena, Parrot, and Peppers. These images are not in our training or cross-validation datasets.

8.2 BIT-RATE SAVINGS

Another way to characterize the artifact reduction system is to measure the equivalent savings in bit-rate. In this approach, we compress a test-set image using divisor table Q_s, and measure the size of the compressed file in terms of bits per color pixel (S_s). We decode the image, apply the artifact reduction system, and measure the perceptual error of the image (E_s). We then use a search technique to find the divisor table KQ_s that results in a compressed image whose perceptual error is equal to E_s, without applying the artifact reduction system. In this search, K is quantized in steps of 0.02, modeling the quantization of this scaling parameter in many JPEG applications. We measure the bits per pixel S_{eq} of the file compressed with KQ_s, and consider the difference $S_{eq} - S_s$ to be the bits gained by artifact reduction. Figure 8 shows this measure, both for the entire test set, and for specific images. We tabulate the average values of S_s, $S_{eq} - S_s$, and the average percentage bit savings, defined to be the average of $(S_s - S_{eq})/S_{eq}$ over the dataset. For the test set, an average percentage bit savings of 14.4% is achieved.

8.3 SCALING PERFORMANCE

We targeted the artifact reduction system to work well for the Q_s quantization tables. In Figure 9, we tabulate the performance of the system on the scaled quantization tables KQ_s, for $K < 1.0$ (higher image quality) and $K > 1.0$ (lower image quality), using the perceptual error, PSNR, and bit-rate savings metrics. We measured this performance because in practice, JPEG end-users often manually scale the Q_s tables to achieve a certain perceptual-quality vs. file size tradeoff, and so a practical JQT would need to work reasonably well for a range of scalings.

Figure 9 shows reasonable performance over the range of K scalings, on all three metrics. The relative performance of the system as a function of K is metric dependent. In terms of bit-rate savings, the system performs best for $K > 1.0$. However, in terms of percent reduction of perceptual error relative to the perceptually Q_e, the system performs best for $K < 1.0$.

8.4 COMPARISON WITH LINEAR NETWORKS

Another way to characterize the artifact reduction system is to compare its performance with a linear system. To perform this comparison, we trained an artifact reduction system that replaced the 64 multi-layer perceptron neural networks with

	Testset	Parrot 768 x 512	Lena 512 x 512	Peppers 512 x 512
Perceptual				
Qs	0.0255	0.0252	0.0362	0.0440
Qe	0.0196	0.0191	0.0298	0.0347
JQT	0.0236 32.2%	0.0221 50.8%	0.0327 55.2%	0.0386 57.8%
Lin: Perc.	0.0248,12%	0.0237, 25%	0.035,24%	0.0420, 21%
Lin: MSE	0.025,10.2%	0.0239, 21%	0.035,24%	0.0425,15%
PSNR				
Qs - Qe	2.50 dB	2.26 dB	1.74 dB	2.20 dB
Qs - JQT	0.634 dB	0.79 dB	0.70 dB	0.68 dB
Lin: Perc.	0.132 dB	0.22 dB	0.18 dB	0.10 dB
Lin: MSE	0.129 dB	0.23 dB	0.21 dB	0.11 dB
bits/pixel				
Ss	1.110	0.517	0.704	0.744
Ss - Seq	0.160 14.4%	0.135 20.7%	0.192 21.4%	0.215 22.4%
Lin: Perc.	0.0503,5.5%	0.051, 9%	0.066,8.5%	0.0682, 8.4%
Lin: MSE	0.0407,4.4%	0.040,7.2%	0.066,8.5%	0, 0%

Figure 8. Tabulation of artifact reduction system performance for the test data set and for three images from the testset that are commonly used in the image processing community; these images are reproduced at the top of the Figure. Results for the baseline nonlinear system and for two linear systems are shown. Three metrics (perceptual error, PSNR, and bits/pixel) are used, grouped in the table by shading; see Sections 7 and 8 for details.

K	0.6	0.8	1.0	1.2	1.4
Percept					
K Qs	0.0208	0.0234	0.0255	0.0273	0.0290
Qe	0.0196	0.0196	0.0196	0.0196	0.0196
JQT	0.0195	0.0217	0.0236	0.0253	0.0265
	111%	46.5%	32.2%	26.1%	26.2%
PSNR					
Qe	0.613dB	1.666dB	2.505dB	3.075dB	3.583dB
JQT	0.504dB	0.577dB	0.634dB	0.648dB	0.645dB
bits/pix					
Ss	1.54	1.29	1.11	1.00	0.897
Ss - Seq	0.170	0.160	0.160	0.140	0.129
	11.4%	12.4%	14.4%	14.2%	14.4%

Figure 9. Artifact reduction system performance for different scalings of the Q_s tables. The shaded $K = 1$ scaling corresponds to the results shown in Figure 8.

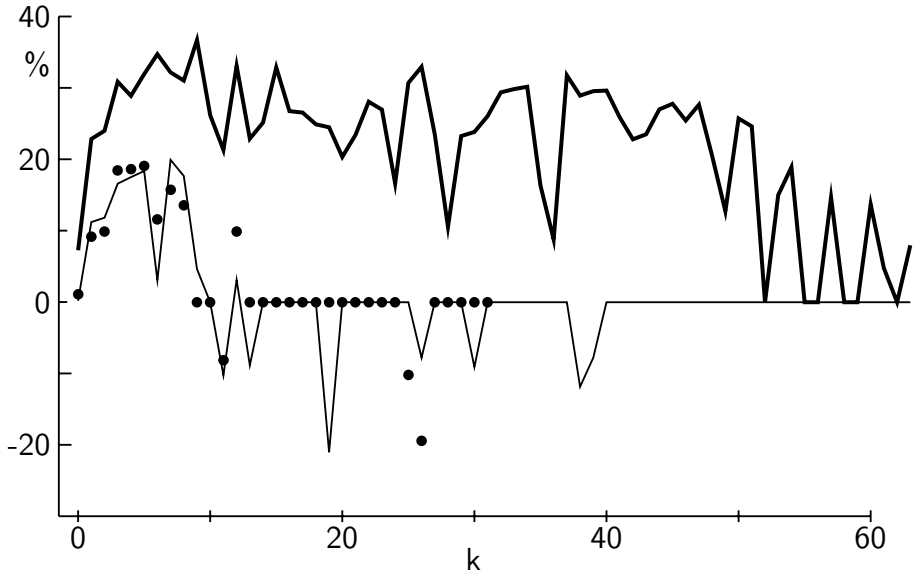


Figure 10. Comparison of the percent reduction of the artifact reduction system (heavy curve, data reproduced from Figure 4) with systems that replaces the multi-layer perceptron neural networks with single-layer linear networks. Thin line shows results from a linear net trained with the perceptual metric; dots show results from a linear net trained with mean square error (computed only for coefficients 0–31).

64 single-layer linear networks. These networks use the same 47 coefficient inputs of architectures A and B (Figure 2a). The output of the linear networks are clipped to the range of plausible values, as indicated by the quantization divisor. We trained the linear networks with gradient descent, using the annealing methods we described in Section 6. Separate linear networks were trained using the constant divisor and constant error methods for each coefficient, and cross-validation performance was used to select the best network. We trained two systems, one using the perceptual error metric to compute the gradient, one using the conventional mean-square error (MSE) metric to compute the gradient.

Figure 10 shows the performance of the linear networks for single-coefficient artifact reduction, using the percentage metric of Equation 5. We reproduced the single-coefficient performance curve shown in Figure 4 for reference. Figure 10 shows the linear networks are markedly inferior in performance for the lowest coefficients, and unable to provide any artifact reduction for all other coefficients. The negative percentages on this graph indicate the poor generalization of the linear networks: the cross-validation results for these coefficients showed improvements over the Q_s error, but test results were inferior to the Q_s error. The linear network trained with the perceptual error metric (thin line) performs better than the MSE-trained networks (dots) on the key low-frequency coefficients 1 and 2. In Figures 5(b) and 5(c) we replot the performance of the linear networks on the two-dimensional u, v coefficient grid, mapping all negative percentages to white (0%).

The single-coefficient results in Figure 10 are confirmed when the linear networks are used for artifact reduction on all 64 coefficients simultaneously. To show the linear networks in the best possible light, we only included networks for coefficients 0-9, eliminating the negative effects of high-frequency poor generalization shown in Figure 10. Figure 8 shows the poor performance of the linear network artifact reduction system, both on the perceptual error measure, PSNR, and bit-savings, both on the full test set data and on selected images. The linear network trained with the perceptual error metric performs marginally better than the network trained with MSE, reflecting the better performance on coefficients 1 and 2 shown in Figure 9.

8.5 IMAGES

Finally, we present color images to qualitatively show the performance of the artifact reduction system (attached to end of paper). Figure 11 shows a closeup of a parrot head in the Parrot image for the 5 values of K tabulated in Figure 9. The top row shows the original image, the upper middle row shows the image compressed with KQ_s (from left to right, $K = 0.6, 0.8, 1.0, 1.2, 1.4$), and the lower middle row shows the results of the artifact reduction system on the KQ_s compressed images. This closeup was chosen to highlight the high-frequency performance of the system: note the corona of artifacts around the parrot's head and beak is significantly reduced by the artifact reduction system. This improvement corresponds to the high performance figures for midrange of coefficients (3–50) in Figure 4.

Figure 12 shows a closeup of the cheek and nose of the Lena image. The format of the Figure is identical to Figure 11. Note the facial discoloration and lip artifacts are modestly reduced at each scaling value. The modest improvement in these low-frequency artifacts corresponds to the modest performance figures for the lowest frequency coefficients (0-2) in Figure 4.

In both Figures, we computed the signed difference between the artifacts present in the upper middle row and the artifacts present in the lower middle row (artifacts were computed by subtraction from the original images). This difference image was scaled (by 3 for Figure 11, by 4 for Figure 12) and added to a neutral gray to produce the bottom row of each image. Non-gray parts of this image indicate areas where the artifact reduction system significantly altered the compressed image; the reader can use this difference image to guide comparisons of the middle rows.

9. DISCUSSION

Examining the results presented in Sections 7 and 8, we see several promising avenues for improving the performance of the artifact reduction system. One avenue concerns improving the accuracy of the perceptual error metric. For example, the current metric does not model chromatic masking or the spatial-frequency dependence of the relative weightings of opponent channel outputs. In addition, the per-coefficient training methods provides an implicit model of spatial-frequency sensitivity, but more explicit modeling of this phenomena may produce better results.

Another promising avenue for further research is improving system performance through more appropriate neural-network architectures. Possible improvements include more hidden layers to model the complexity of artifacts, and an automatic method for choosing inputs relevant for each coefficient. These improvements need to focus on the lowest frequency coefficients, where the current system shows only modest performance improvements.

Apart from performance improvements, the work presented in this paper requires other enhancements in order to be used in a practical system. A practical JQT implementation must also include a method of requantization. As implemented in this paper, the enhanced coefficients $\tilde{k}^C(u,v)$ are maintained as floating point values. To create the transcoder JPEG file, a JQT must decide, for each enhanced coefficient, how many bits of precision should be maintained. This decision involves both the importance of the coefficient to the quality of the image, and the confidence that the enhanced coefficient results in less artifacts than the original coefficient.

10. CONCLUSIONS

We have presented a neural network image processing system that operates directly on a compressed representation of an image, and that uses a perceptual error metric to guide supervised learning on a large image database. We believe this approach has more general application in image processing, beyond the artifact reduction problem.

An advantage of this approach is the ability to define a variant of a general problem by customizing the training database of images. A JQT customized for photographs of faces can be specified by assembling an image database with a heavy representation of these types of photos. A JFIF transcoder that corrects for artifacts caused by an inexpensive analog-to-digital conversion in a consumer digital camera can be trained by collecting a database using a prototype camera that has auxiliary high-quality conversion circuitry. This ease of customization may be the deciding factor for using a pattern recognition approach for a particular problem in digital imaging.

Acknowledgments

We thank Krste Asanovic, Dan Hammerstrom, Yann LeCun, Richard Lyon, John Platt, Larry Yaeger, and the reviewers for the journal version of this technical report for useful comments. Funded by DARPA DABT63-96-C-0048.

APPENDIX 1: YC_rC_b to LMS TRANSFORMATION

In this section, we derive the transformation from YC_rC_b color space to LMS color space. The YC_rC_b color space, as used in the JPEG/JFIF standards, uses 8-bit integer values. Substituting this scaling into the suggested YC_rC_b to RGB conversion in CCIR Recommendation 601-1 yields:

$$R = (Y/255) + 1.402(C_r/255)$$

$$G = (Y/255) - 0.3441(C_b/255) - 0.7141(C_r/255)$$

$$B = (Y/255) + 1.772(C_b/255).$$

These equations assume Y ranges from 0 to 255, and C_r and C_b range from -128 to 127. The R, G, and B values are valid between 0.0 and 1.0. Since some values of YC_rC_b may produce RGB values outside the valid range, we clamp the RGB values to a maximum of 1.0 and a minimum of 0.0.

Following CCIR Recommendation 709 (D65 white point), we convert RGB to 1931 2-deg CIE XYZ tristimulus values using the equations:

$$X = 0.4124R + 0.3576G + 0.1804B$$

$$Y = 0.2127R + 0.7152G + 0.07217B$$

$$Z = 0.01933R + 0.1192G + 0.9502B.$$

We use the following equations, derived for typical spectral power distributions of the phosphors in a Sony 17 inch color monitor (Tjan, 1996), to convert CIE 1931 XYZ values to Judd-Vos tristimulus values $X_pY_pZ_p$:

$$X_p = 0.9840X + 0.00822Y - 0.00459Z$$

$$Y_p = 0.00028X + 0.9992Y + 0.00519Z$$

$$Z_p = -0.00177X + 0.00388Y + 0.9215Z.$$

To complete the transformation to LMS space, we convert Judd-Vos tristimulus values to Smith-Pokorny cone excitations (Tjan, 1996):

$$L = 0.1551X_p + 0.5431Y_p - 0.03286Z_p$$

$$M = -0.1551X_p + 0.4568Y_p + 0.03286Z_p$$

$$S = 0.00801Z_p.$$

These operations can be collapsed into two sets of linear equations and a clipping operation. Appendix 2 includes this compact form of the YC_rC_b to LMS transformation.

APPENDIX 2: PERCEPTUAL ERROR METRIC

In this section, we define a pointwise perceptual metric, computed on a pixel (Y, C_b, C_r) in an original image and the corresponding pixel $(\hat{Y}, \hat{C}_b, \hat{C}_r)$ in a reconstructed image. We begin by converting both points from YC_bC_r color space to LMS space, as derived in Appendix 1. We assume Y ranges from 0 to 255, and C_b and C_r range from -128 to 127.

$$R = 0.003922Y + 0.005498C_r$$

$$G = 0.003922Y - 0.001349C_b - 0.002800C_r$$

$$B = 0.003922Y + 0.006949C_b.$$

Clamp R, G, and B to lie between 0.0 and 1.0, then compute LMS values as:

$$L = 0.17816R + 0.4402G + 0.04005B$$

$$M = 0.03454R + 0.2750G + 0.03703B$$

$$S = 0.0001435R + 0.0008970G + 0.007014B.$$

Using (L, M, S) and $(\hat{L}, \hat{M}, \hat{S})$ values, we compute cone contrast vectors as:

$$\Delta L/L = \frac{L - \hat{L}}{L + L_o}$$

$$\Delta M/M = \frac{M - \hat{M}}{M + M_o}$$

$$\Delta S/S = \frac{S - \hat{S}}{S + S_o}.$$

Using the recommendations in (Macintyre and Cowan, 1992) for the dark pixel characteristics of commercial CRT monitors, we set $L_o = 0.01317, M_o = 0.006932, S_o = 0.0001611$. Consult (Macintyre and Cowan, 1992) for details on tuning these values to a particular CRT monitor. Using these cone contrast values, we compute opponent space values as:

$$BW = A(x, y)(\Delta L/L + \Delta M/M)$$

$$RG = \Delta L/L - \Delta M/M$$

$$BY = \Delta S/S - 0.5(\Delta L/L + \Delta M/M)$$

where A(x,y) is the activity function value for the pixel position under comparison. Using these opponent values, we compute the error function

$$E(Y, C_b, C_r; \hat{Y}, \hat{C}_b, \hat{C}_r) = 0.07437\,|BW| + 0.8205\,|RG| + 0.1051\,|BY|.$$

The channel weightings in the error function are the averaged relative sensitivities of the three subjects measured in (Cole et al, 1993). The error function sums the absolute values of the opponent channels, rather than the squared values used in a Euclidean metric: this choice reflects the assumed independence of the underlying mechanisms.

Straightforward application of the chain rule yields the partial derivatives $\partial E()/\partial \hat{Y}$, $\partial E()/\partial \hat{C}_b$, $\partial E()/\partial \hat{C}_r$ used in the backpropagation learning algorithm.

To compute the $A(x, y)$ function over the original image, we use the Y component of YC_bC_r pixels directly, without converting to LMS space. We take this approach because BW in opponent space and Y in YC_bC_r space are qualitatively similar measures of luminance.

To compute $A(x, y)$, we first compute the mean luminance value Y_m in a 5 by 5 block centered around pixel position (x,y). We then compute the contrast Y/Y_m for each pixel in a 5 by 5 block centered around pixel position (x,y), and clamp the lower limit of this contrast at $10/Y_m$. If a contrast is less than 1, we take its reciprocal. We sum these 25 modified contrast values, divide by 25, and take the reciprocal of the result to produce $A(x, y)$. The function is an average measure of edge activity in a region, that takes a value of 1 for smooth areas, and a value less than one for regions around an edge.

References

Ahumada, Albert J. and Horng, Rensheng (1994). Smoothing DCT compression artifacts. *In 1994 SID International Symposium Digest of Technical Papers.* p. 708-11.

van der Branden Lambrecht, Christian J., and Farrell, Joyce E. (1996). Perceptual quality metric for digitally coded color images. *Proceedings of EUSIPCO-96,* Trieste, Italy.

Cole, Graeme R., Hine, Trevor, and McIlhagga, William (1993). Detection mechanisms in L-, M-, and S-cone contrast space. *Journal Optical Society of America,* Vol. 10, No. 1.

Fuhrmann, Daniel R., Baro, John A., and Cox, Jerome R. (1995). Experimental evaluation of psychophysical distortion metrics for JPEG-encoded images. *Journal of Electronic Imaging,* Vol. 4, No. 4, pp. 397-406.

Jarske, Tiina, Haavisto, Petri, and Defee, Irek (1994). Post-Filtering Methods for Reducing Blocking Arifacts from Coded Images. *IEEE Transactions on Consumer Electronics,* Vol. 40, No. 3, pp. 521-526.

Kim, Kyeong Man, Lee, Chae Soo, Eung, Joo Lee, and Yeong, Ho Ha. (1996). Color Image Quantization and Dithering Method based on Human Visual System Characteristics. *Journal of Imaging Science and Technology,* Vol. 40, No. 6, pp. 502-509.

Macintyre, Blair and Cowan, William B. (1992). A Practical Approach to Calculating Luminance Contrast on a CRT. *ACM Transaction on Graphics,* Vol. 11, No.

4, pp. 336–347.

Minami, Shigenobu and Zakhor, Avideh (1995). An Optimization Approach for Removing Blocking Artifacts in Transform Coding. *IEEE Transactions on Circuits and Systems for Video Technology,* Vol 5, No 2., pp. 74-81.

O'Rourke, Thomas P. and Stevenson, Robert L. (1995). Improved Image Decompression for Reduced Transform Coding Artifacts. *IEEE Transactions on Circuits and Systems for Video Technology,* Vol 5, No 6., pp. 490-499.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning Internal Representations by Error Propagation. In Rumelhart, D. E. and McClelland, J. L. (eds), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations.* MIT Press.

Tjan, Bosco S. (1996). Color Spaces for Human Observer. Technical Memo, Minnesota Laboratory for Low-Vision Research, University of Minnesota. Available online at `http://vision.psych.umn.edu/www/people/bosco/techs.html`.

Wallace, Gregory K. (1992). The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics.* Vol 38 No 1 pp. 18-34.

Westen, S. J. P., Lagendijk, R. L., and Biemond, J. (1996). Optimization of JPEG color image coding using a human visual system model. *Proceedings of the SPIE,* Vol. 2657, pp. 370-81.

Wu, Siu-Wai and Gersho, Allen (1992). Improved Decoder for Transform Coding with Application to the JPEG Baseline System. *IEEE Transactions of Communications,* Vol 40, No 2., pp 251–254.

Yang, Yongyi, Galasysanos, Nikolas P., and Katsaggelos, Aggelos K. (1995). Projection Based Spatially Adaptive Reconstruction of Block-Transform Compressed Images. *IEEE Transactions on Image Processing,* Vol. 4 No 7., pp. 896-908.

Color Plate Captions

Figure 11. Closeup from Parrot image, showing high-frequency system performance at 5 scalings of Q_s. From left to right, scalings are $0.6Q_s$, $0.8Q_s$, $1.0Q_s$, $1.2Q_s$, $1.4Q_s$. From top to bottom: original image, image compressed with KQ_s, image compression with KQ_s followed by artifact reduction system, difference image (see text for details).

Figure 12. Closeup from Lena image, showing low-frequency system performance at 5 scalings of Q_s. Format identical to Figure 11.