Improving Wavelet Image Compression with Neural Networks

Christopher J.C. Burges, Patrice Y. Simard,
and Henrique S. Malvar
{cburges,patrice,malvar}@microsoft.com

Technical Report
MSR-TR-2001-47

We explore the use of neural networks to predict wavelet coefficients for image compression. We show that by reducing the variance of the residual coefficients, the nonlinear prediction can be used to reduce the length of the compressed bitstream. We report results on several network architectures and training methodologies; some pitfalls of the approach are examined and explained. A two layer fully connected network, trained offline, applied to a test set consisting of seven 512 by 768 images from the Kodak database, gives a consequent overall improvement in the bit rate of between 4% and 7%.

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

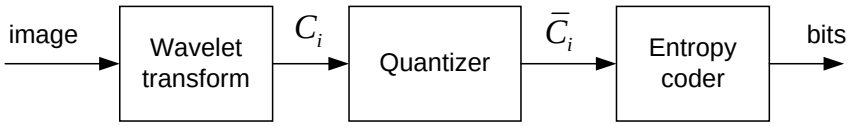

Figure 1: Simplified block diagram of a typical wavelet signal coder.

1 Introduction

Most wavelet-based signal compression systems [1] are based on the structure shown in Figure 1. The wavelet coefficients C_i are quantized (divided by a step size and then rounded to nearest integers), and the resulting indices (noted $\bar{C}_i$ in the figure) are encoded without loss by the entropy encoder box, which usually employs contextual information. Higher amounts of compression are obtained by increasing the quantization step sizes (so that quantized values equal to zero are more likely), and by making better prediction for the ranges of quantized values via appropriate contexts and data structures.

For example, in the PWC (progressive wavelet coder) image encoder [2], entropy coding performance is increased through the combination of two steps: "ping-pong" wavelet coefficient reordering (which tends to cluster coefficients quantized to zero), and adaptive Run-Length-Rice coding. In the traditional embedded zerotree wavelet (EZW) scheme [3], context modeling is performed by predicting the coefficient variance of a pixel from the pixels corresponding to the same spatial locations but lower resolution subbands.

Suppose that the $\bar{C}_i$ are independent and normally distributed. Then for a large number of quantization bins, the entropy H (and hence the number of bits required to encode a given image) scales with the standard deviation σ as $H \sim \log_2(\sigma)$. This motivates the proposed coder architecture, which attempts to reduce the variance of the residuals by subtracting the predicted values, which are output from a nonlinear predictor.

One could attempt prediction of pixel values. However, predicting wavelet coefficients, and training a separate neural net for each wavelet level and subband, has the advantage that the nets do not have to learn scale information. In Figure 2 we show a typical wavelet subband decomposition. The notation L and H stand respectively for lowpass and highpass filtering (the first letter denotes vertical and the second horizontal directions). The LL wavelets themselves work as smooth predictors (lowpass filters) and the LH, HL, and HH coefficients are the residuals computed from these predictors. However, because the wavelet decomposition is linear, nonlinear dependencies among wavelet coefficients still remain. The neural net prediction proposed in this paper exploits such nonlinear dependencies to improve prediction.

In this paper, we will use PWC [2] with 9/7 biorthogonal wavelets [1] as the baseline system, in order to gauge the effects of prediction on a known, end-to-end system. PWC has been shown to give results comparable to the state-of-the-art SPHT codec

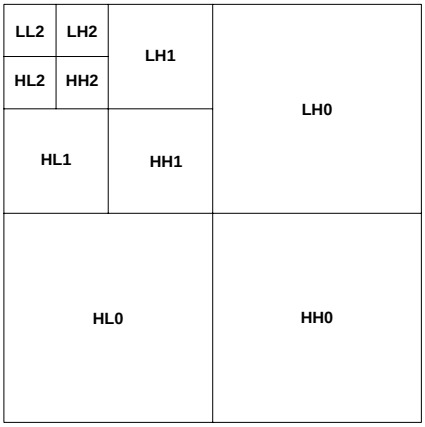

Figure 2: Multi-resolution wavelet representation (three levels shown).

[4]. Figure 3 shows how the proposed nonlinear predictor fits into the PWC codec architecture. (For simplicity, the predictor box has its own quantizer and unquantizer).

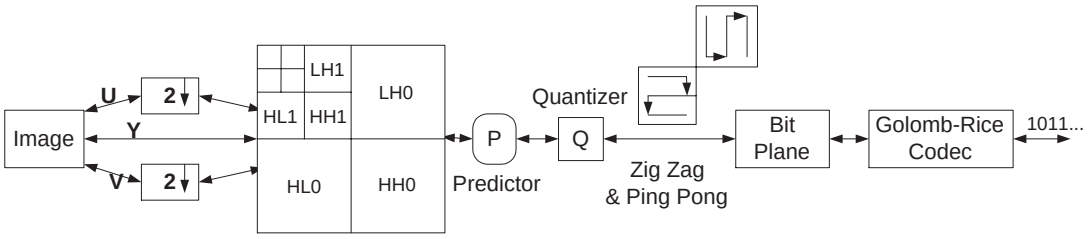

Figure 3: The PWC codec [2], with nonlinear prediction added.

Section 2 outlines how a general nonlinear predictor can be used, and Section 3 discusses the neural network predictor in particular. Section 4 presents experimental results verifying the compression gain attained with nonlinear prediction and describes its computational complexity. Final remarks are given in Sections 5 and 6.

2 Using A Nonlinear Predictor

The predictor block in Figure 3 is broken down in Figure 4. There, the C_i are the wavelet coefficients, P_i the predicted values for the wavelet coefficients, $R_i = C_i - P_i$ the residuals (with $\bar{R}_i$ and $\tilde{R}_i$ the corresponding quantized and then unquantized values), and the $\tilde{C}_i = P_i + \tilde{R}_i$ are the reconstructed wavelet coefficients. $\tilde{C}_{j<i}$ denotes a fixed number of previously encoded wavelet coefficients, used as inputs to the nonlinear predictor. In the corresponding decoder (bottom half of Figure 4), the predictor performs exactly the same operations, using the same contexts (built from previously reconstructed wavelet coefficients) as the predictor in the encoder. Note that the prediction is done solely using local data; no side information is sent.

Since the problem we are solving is essentially a noisy regression problem (although as we will see, it has a dynamical aspect too), a variety of techniques could be used,

such as support vector machines [5] or generalized linear models [6]. We chose to use a neural network since its computational cost can be directly controlled and training is straightforward.

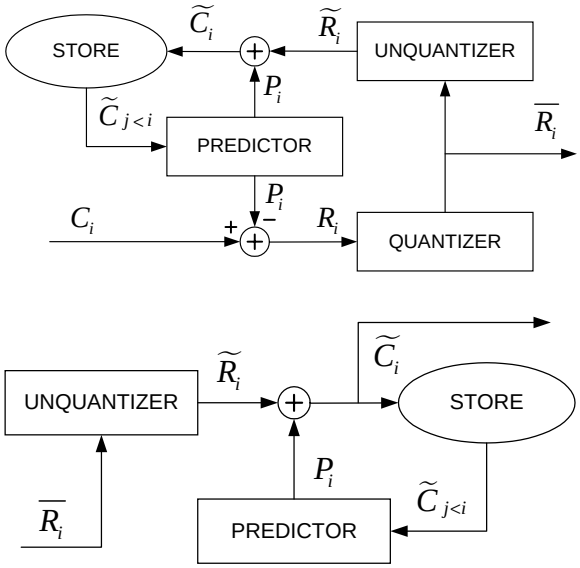

Figure 4: Block P of Figure 3, expanded. Top: Proposed encoder, with an additional nonlinear prediction step. Bottom: the corresponding decoder. Note that the predictor is in a closed loop.

3 The Neural Network Predictor

We consider a two layer neural network with a single output $F(\cdot)$, parameterized by the vector of weights W. The output unit is a sigmoid taking values in $[0, 1]$; a network is trained for each subband and each wavelet level, and the outputs are translated and rescaled, again per each subband and wavelet level. Similarly, the inputs are rescaled so that most lie in the interval $[-1, 1]$. Since the goal is to train the net in order to minimize the variance of the prediction residuals $R_i = C_i - F(C_{j<i}, W)$, we use the mean-squared error measure:

$$E(W) \equiv \sum_i R_i^2 = \sum_i (C_i - F(C_{j<i}, W))^2 \tag{1}$$

Note that, assuming that quantization errors are not excessively large, since the net sees reconstructed wavelet coefficients as inputs, we can train the net on the original wavelet data using the above error measure, even though the net will operate on quantized data. The advantage of doing this is that training does not depend on the quantization value Q. Below we will describe both training using static wavelet coefficients, and training using coefficients computed dynamically with the network in the loop; the latter method does depend on Q.

3.1 Training

In our proposed coder, the net is fed a causal context as input and aims to predict the next wavelet coefficient. In batch learning, Eq. (1) is computed using the whole train set, and the weights are then accordingly updated. In stochastic gradient descent learning, one pattern at a time is presented to the network, with the corresponding error $E_i(W)$ given by

$$E_i(W) = (C_i - F(\tilde{C}_{j<i}, W))^2 \tag{2}$$

and the overall error is reduced by updating the weights according to

$$W^{t+1} = W^t - \epsilon \frac{\partial E_i(W)}{\partial W} \tag{3}$$

where ϵ is the learning rate. Stochastic gradient descent gives similar results and is usually preferred if the training set is large and redundant, and we use it here. However there remain several possible training methodologies. One could start with an untrained net for each image and train on the fly, using the reconstructed coefficients for the current image as data. The advantage is that the network can adapt to the statistics of the current image, but at the price of increased processing time. Also there must be sufficient data in the image for the learning to converge sufficiently well to be useful; a poor predictor will actually hurt performance. Convergence could be improved by starting with a net trained on a fixed set of train images; however, the speed disadvantage remains. Alternatively, one could train on each specific image, and send W as side information. We can also train the network to minimize its number of weights using the "optimal brain damage" algorithm [7]. For our coder, we estimate that the side information overhead would be less than 2%. Finally, we can train on a fixed set of images, and make the (fixed) network weights a part of the codec for all images. It is not *a priori* obvious that this will work well, since it assumes that the net can learn nonlinear dependencies of the wavelet coefficients that apply to all, or most, images. However experiments showed that the fixed-weights method worked better than the on-the-fly training, and has the significant advantage that it does not require that the codec perform training, which is prohibitively slow.

3.2 Context and Network Architecture

The context $\tilde{C}_{j<i}$ used to predict the ith coefficient should contain as few coefficients as possible, to minimize complexity. Still, it should include enough coefficients in order to exploit statistical dependencies. Although earlier wavelet codecs such as EZW [3] attempted to exploit dependencies among wavelet coefficients across levels (scales), recent codecs such as PWC [2] have similar performance while exploiting only dependencies within a level. A more formal analysis in [8] also recommends that interscale dependencies do not help much if intrascale ones are fully exploited. Experiments with 3 by 3 parent contexts in the LH0 band, used in a predictor for HL0, did not show any gains. Therefore, we chose to use a context of neighboring pixels within the same level (cf. Figure 2).

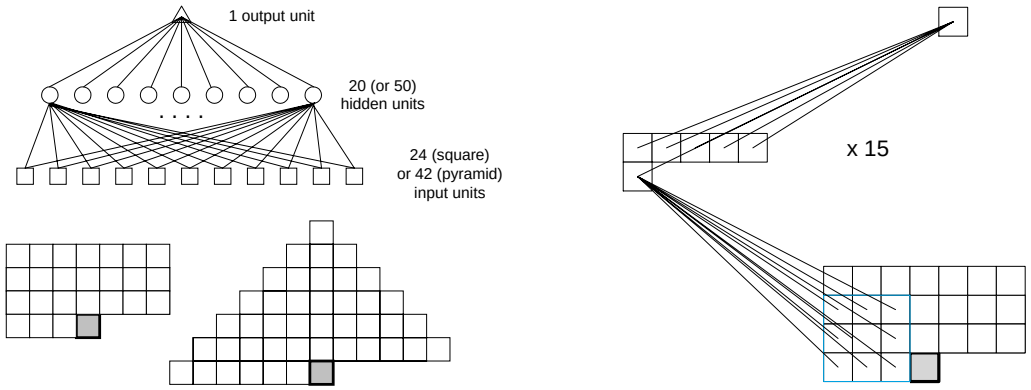

Figure 5: Top left: simplified diagram of the fully connected neural network predictor. Bottom left: contexts that generate the inputs to the net: box and pyramidal. Right: The convolutional net. One of the 15 hidden unit feature vectors is shown. The last layer is fully connected.

We considered two basic network structures: fully connected nets, and convolutional nets [9]. The fully connected net architectures are shown on the left in Figure 5. The number of inputs is equal to the context size, and several different numbers of hidden nodes were investigated. Two kinds of causal context were examined: a box shaped one, and a pyramidal one (left panel of Figure 5). The idea behind the pyramidal context is that if the influence of some wavelet coefficient on the value to be predicted depends mainly on the Euclidean distance between the two (in the corresponding subband and wavelet level), one should use a radial context.

The convolutional net has two advantages, at least when used on images: the convolutional kernels learn what features are important in the data (for example, on pixel data, some usually become edge detectors), and when replicated across an image, a convolutional net can yield a significant computational savings [9] (however, other options for improving speed, some of which are described below, are no longer possible)[1]. We chose the architecture shown in the right panel of Figure 5. There, one of the 15 feature vectors is shown, and each square corresponds to a convolutional kernel with 9 fixed weights; the feature vector is populated by replicating the kernel across the inputs. Since each kernel becomes a feature detector, we chose 15 kernels, giving 90 hidden units. Using many more would have nullified the speed advantage. Although the chosen architecture has 900 weights (and 225 free parameters), if the net is shifted one column to the right in the array of coefficient values, 225 new multiply adds are required to compute the network output (assuming that all previously computed values are stored), compared with 500 for a fully connected net with 20 hidden units.

[1]Note that all the methods considered involved convolving a network across the whole array of wavelet coefficients. Here we are referring to a convolution within a context.

4 Experimental Results

We performed experiments in two stages, to save time: initially, we used the first 5 images from the Kodak database as train data, the 6th as validation data, and image 7 through 11 as the test set; final results are then reported using the first 16 images as train, the next as validation, and the next 7 as test images. We only performed experiments on the Y values of the YUV decomposition, since most of the information resides in those coefficients.

Initial experiments showed that using the net trained for LH0, on the transpose of the data in HL0, gave similar results to training a net specifically on transposed HL0 data, and so we used the LH0 net for both subbands. We also checked that given the architecture in Figure 5, taking the transpose of the LH0 data first and then training the LH0 net, did not give as good results.

4.1 Training Data

In the first round of experiments training data was generated statically, that is, for a given subband and wavelet level, the causal context for each coefficient to be predicted was computed and saved, and corresponding data from all train images concatenated and then shuffled (the latter significantly helps speed training convergence). For LH0 this generated a large quantity of data (589,824 train patterns), so to speed training we took only the first 150,000 patterns of the shuffled database; experiments using 250,000 patterns showed no improvement in generalization performance, indicating that the capacity of the network was suited to the smaller amount of data.

The variance of the coefficients varies significantly from one subband and wavelet level to another, and tend to decrease rapidly as we move from high levels towards level 0 [3]. Table 1 shows the variance of various coefficients in the 16-image train set for the first three wavelet levels. We rescaled and translated the coefficients for input to the net by requiring that 7 standard deviations map to the interval $[-1, 1]$. The net outputs were then rescaled from $[0, 1]$ to the same range. For a given network, the rescaling of the inputs is in fact not necessary, since the weights will eventually adjust to compensate for the chosen scale factor, but mapping to $[-1, 1]$ is likely to improve the convergence rate and avoid numerical problems. Scaling the outputs is also necessary, given that we used a sigmoid transfer function in the output unit. We explored circumventing the output scaling factors by using a linear transfer function instead, but these networks gave significantly worse generalization performance. The problem is that since the wavelet coefficient distribution is long-tailed (in fact approximately Laplacian), the network will be presented with large valued targets during stochastic training, and the corresponding large error values will bias the gradient descent, giving the examples with large target values undue weight. Using a sigmoidal output unit solves this, since the error value will be at most one.

LH0	66	HL0	54	HH0	18
LH1	418	HL1	411	HH1	190
LH2	2217	HL2	1973	HH2	761

Table 1: Variance of wavelet coefficients for first three wavelet levels.

4.2 Choice of Context

We trained a network with pyramidal context on the first five images, with 50 hidden units and pyramid base of size 11 (see Figure 5). The net has 42 (=6+11+9+...+1) input units and a total of 2150 weights. We compared performance with a 7+7+3 box context, and a 7+7+7+3 box context, each with 20 hidden nodes (giving 360 and 500 weights, respectively). Comparisons were made by fixing the peak signal-to-noise ratio (PSNR) [2] over each test image to 36 dB for all experiments (experiments with different PSNRs lead to similar results). Since most of the coefficients are at wavelet level 0, for this experiment we only trained a predictor for the LH0 subband. Averaged over the 5 test images, the pyramid net gave a 1.1% overall reduction in the number of bits, the smaller box context net ('box net') 1.0%, and the larger box net 1.5%. Based on these results, and the fact that the larger box net is 4 times faster than the pyramid net in test phase, we used the former for the remainder of the experiments. The worse performance of the larger network is likely due to excess capacity.

4.3 Choice of Network Architecture

The two architectures we examined were fully connected networks and a convolutional net. However using the convolutional architecture described in Section 3.2 for the LH0 data did not lead to improved generalization performance over the box net: it gave 0.9% compression on the 5 test images, as opposed to 1.5% for the box net. The convolutional net we tested has fewer free parameters than the box net (225 vs. 500). The observed drop in accuracy may be due to the lower number of free parameters; however the kinds of features present in the wavelet coefficients are very different (e.g. have much higher variance) than those found in images, and this is likely to have also contributed. For example, convolutional kernels are good for building approximate translation invariance into the system when used on pixel data [9], but due to the critical sampling of the wavelet transform used, small translations of the image can generate large differences in wavelet coefficients, and so one cannot expect the same advantage here.

4.4 Training Results

Focusing on the best performing box nets, we trained nets with 5, 10, 20, 30 and 40 hidden units (all with the same 7+7+7+3 box context input layer). The motivation was twofold: to match the capacity of the network to the available data; and to gain the computational savings from using a smaller network where possible. The

resulting percentage decrease in the minimum squared error on the validation image are shown in Table 2. Columns for 30 and 40 hidden units are not shown because the corresponding values were within 1% of the numbers for 20 hidden units. Note that the amount of train data drops by a factor of 4 as we go up each wavelet level: the number of train patterns for the LH0 net is 1.6 million, and that for the LH2, 98,304. We see from the Table that the reduction in mean squared error is beginning to decrease as the number of hidden units drops to 5, but that it is otherwise quite insensitive to the size of the hidden layer.

	$\epsilon = 0.1$			$\epsilon = 0.01$		
	5	10	20	5	10	20
LH0	11	14	15	12	14	14
HH0	7	8	8	8	8	8
LH1	8	9	9	8	9	9
HH1	0	0	0	1	0	0
LH2	12	13	13	10	10	10
HH2	0	0	0	0	0	0

Table 2: Percentage reduction in mean squared error on the validation image after training; ϵ is the learning rate (cf. Eq. 3).

4.5 Higher Wavelet Levels: Network Oscillations

Application of the above prediction scheme to higher wavelet levels (lower resolution) uncovered an interesting problem. Because the net outputs are used to construct inputs for subsequent application of the same net, feedback can occur. This can lead to spurious responses, much like limit-cycles in quantized digital filters. To illustrate this, we used a box net trained for LH2, setting the quantization threshold so high that all wavelet coefficients quantize to zero. Thus the reconstructed wavelet coefficients are due purely to the previous network outputs, and since the network sees its own outputs cumulatively, significant spurious predictions can result. The reconstructed pixel result is shown in Figure 6, where the effect has been ampilified visually by scaling values so that all data within two standard deviations of the mean map to the dynamic gray level range. The bands are purely a consequence of feedback. Some reconstructed images show similar but much less noticable distortions. Clearly this effect introduces noise in the process and will impact the rate distortion curve detrimentally. The question arises: would an arbitrary net (e.g. one not trained on the wavelet data) give a similar effect, or is the effect the result of learning? To answer this, we used a network with random weights for prediction again of LH2 only. For weights in $[-0.1, 0, 1]$ we indeed observed a similar reconstructed image (after again mapping gray levels to the dynamic range). However for weights in $[-1, 1]$ we observed the pattern shown in Figure 6 (right panel).

In order to solve this problem, we must adjust the training of the network so that

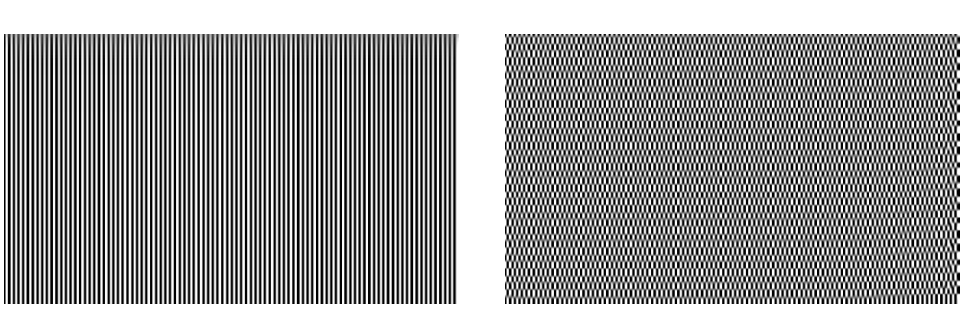

Figure 6: Spurious output patterns generated with zero inputs, for a trained net (left) and a net with random coefficients (right).

such effects are penalized. To do this, instead of training on static data, we use the causal contexts of reconstructed wavelet coefficients computed on the fly, using the network as it trains. In this way, any feedback is included in the training itself, so it contributes to the mean squared error seen by the network, and is thus penalized. Using this method solved the problem: for the high quantization experiment described above, the error produced by the feedback loop (i.e. the standard deviation of the reconstructed coefficients) was reduced by a factor of over 100 (and the mean value reduced from 8.4 to -0.1), and no visible artifacts occurred in the reconstructed images.

Using this method of training has two other side effects: first, the training data itself is now dynamic, and shuffling the data is no longer desired. Second, whereas before the nets could be trained on the original wavelet coefficients, now one must choose a quantization level used during the training process. We chose a small value of $Q = 10$, so that the resulting distortion is small (and the nets see quantities that are close to the original coefficients). Finally, since the input data now changes with time, rather than stopping training after a fixed number of iterations, we stop after 100 iterations have occurred during which no further reduction of the mean squared error on the validation image has been achieved. The saved net is that for which the validation error is minimized. The experiments described in the next two sections used this method of training.

4.6 Variance, Entropy and Quantization

Histograms of wavelet coefficients are strongly peaked [8], and they concentrate even further after prediction. A typical example is shown in Figure 7, for the LH0 subband in one of the test images. In this case, as expected the variance decreases, and thus a compression gain results. However at relatively low coding rates, reduction in variance may not lead to as much reduction in entropy as the high-rate asymptote $H \sim \log_2(\sigma)$.

To investigate the effect of residual variance reduction on entropy, we trained box nets for subbands LH0 through LH2, using the full train set (the first 16 images). We then computed the variance and entropy, averaged over all train images, for subbands and wavelet levels LH0, LH1 and LH2. Here, by "entropy" we mean the empirical first order entropy of the resulting, quantized residuals. Since we are interested in the

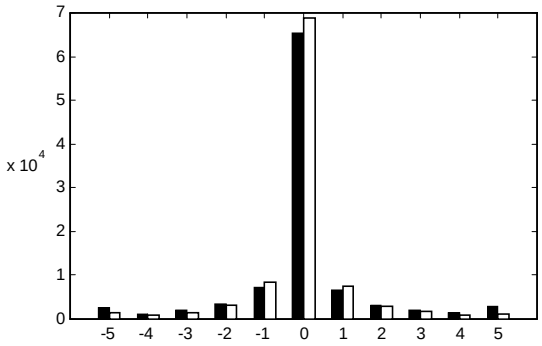

Figure 7: Typical histogram of quantized wavelet coefficients (black) and prediction residuals (white).

effect of bin size on the variance and entropy, we computed these quantities for values of the scalar quantization level Q [2] from 1 to 100 in integer steps. The resulting curves for residual entropy and variance are shown in Figure 8. Note that the drop in entropy at low Q values is much larger for wavelet level 0 than for the higher levels. In fact, for LH2, for quantization step sizes between 10 and 30, there is very little reduction in mean entropy, despite the reduction in variance.

Figure 9 shows the corresponding LH2 graphs for images # 1, 5, and 15 in the train set. We see that sometimes the predictor helps, but sometimes it can degrade performance. The first image, for which both variance and entropy are reduced, has many vertical and horizontal edges, and performance is improved; the fifth, for which both are increased, has edges, but few vertical and horizontal ones, and performance is actually degraded; and the fifteenth, for which the variance is reduced (at low Q) but the entropy not, has few sharp edges. A reduction in variance which is not accompanied by a reduction in entropy usually means that at higher wavelet levels the variance of the coefficients is much higher, so the nonempty bins after quantization are more widely spread. Thus, the variance can be reduced significantly if whole bins move towards the mean, without a significant effect on the entropy.

4.7 Complexity Results

Based on the above observations, in the following experiments we chose to use prediction only on wavelet levels 0 and 1, and for subbands LH, HL and HH, and we used the fully connected net with box context inputs. Throughout, 5 levels of wavelet levels were used altogether. We used quantization level $Q = 10$, which gives a PSNR such that any distortion is hard to discern visually (usually between 36 and 40). For the architectures considered, with N hidden nodes, the number of multiply add operations required for a single evaluation is $25N$. Thus for the 20-hidden node network, the number of multiply adds, *per image pixel*, required to incorporate the prediction, is approximately 470. Since this computational cost applies to both the encoder and the decoder, this is a drawback of the proposed approach for applications where real time performance is important. However there are several possible ways to significantly

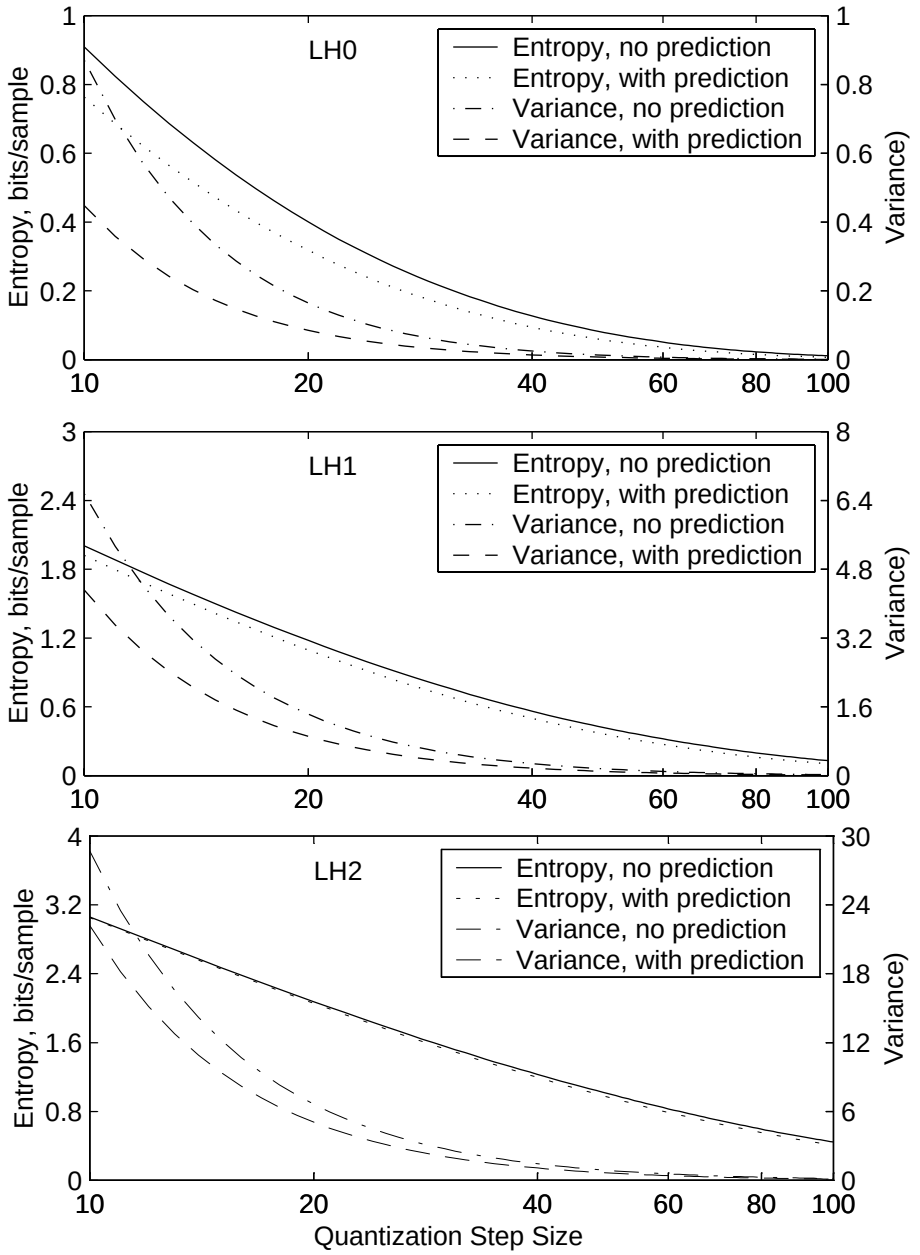

Figure 8: Effect of neural net prediction on the variance and entropy of subbands LH0 (top), LH1 (middle), and LH2 (bottom), averaged over the training set.

improve the speed, which we now explore. The first one we call a "context-test": if the network inputs that are contiguous with the coefficient to be predicted (see Figure 5) become zero after quantization, we simply predict 0 (i.e. don't call the net). For the box causal context used here, this requires testing 4 coefficients. The number of calls to the network without this scheme is fixed at 368,640 (294,912 and 73,728 for wavelet levels 0 and 1, respectively) per image. Table 3 shows the percentage reduction in the mean number of calls to the network, over each test image, for wavelet

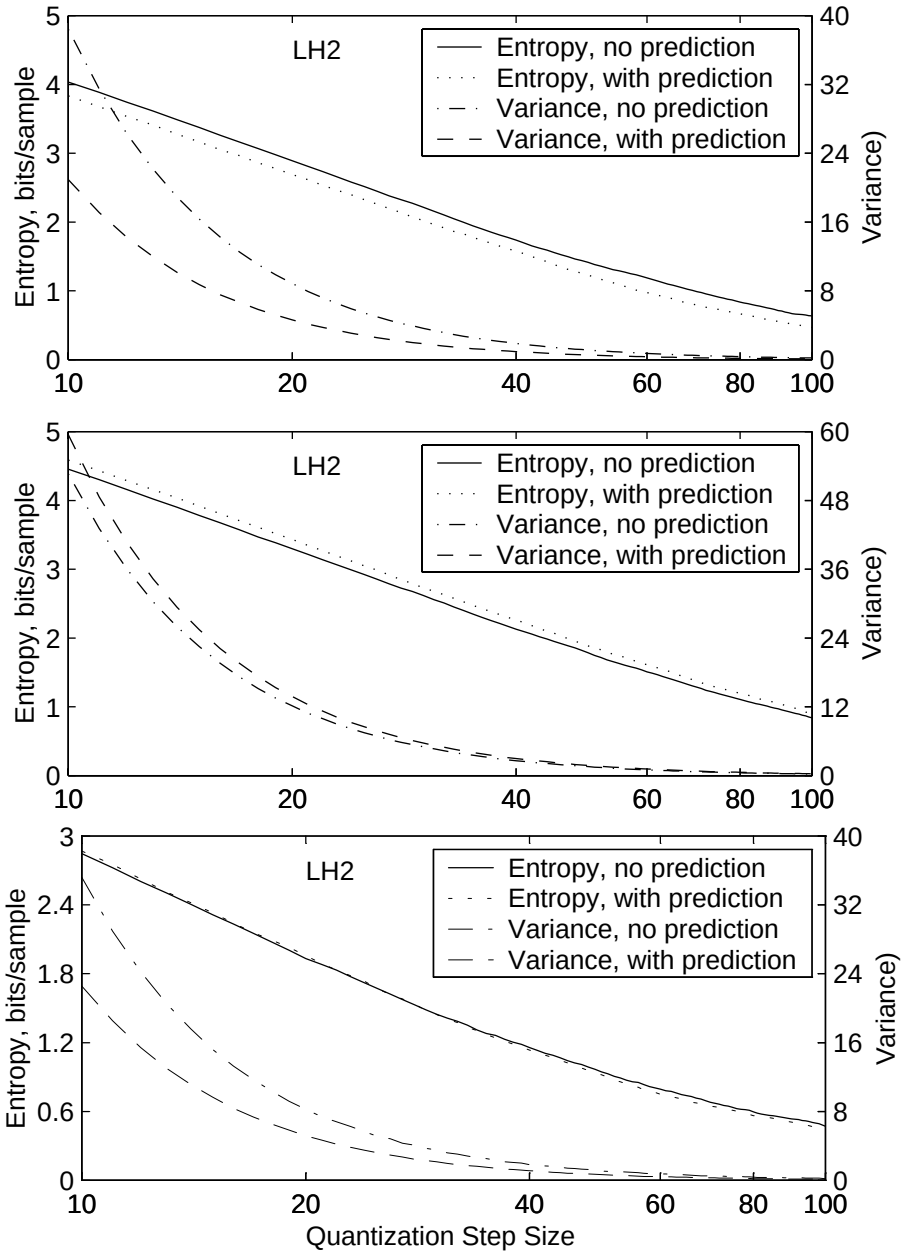

Figure 9: Effect of neural net prediction on the variance and entropy of subband LH2, for three of the training images.

levels 0 and 1, with the scheme implemented[2].
The mean reduction in the number of calls, over all test images, is 77% for wavelet level 0, and 49% for wavelet level 1. This speedup method was found to actually improve performance slightly, as we will show in the next Section[3].

[2]The different networks had slightly different numbers of calls due to the feedback loop.

[3]The method does have some overhead in computing whether the inputs would quantize to zero. However this is small (and constant time) since one only has to do it once for each wavelet coefficient.

Test Image	Wavelet Level 0	Wavelet Level 1
18	68.7	28.7
19	74.3	50.4
20	83.8	64.7
21	72.1	50.8
22	76.9	39.7
23	94.6	79.2
24	66.9	31.9

Table 3: Percent reduction in number of calls to the neural network for each test image, using the 4-coefficient context test.

Another speedup method takes advantage of the sparseness of the network's inputs: given that the causal context has passed the above test, only forward propagate those values in the network's inputs which would not quantize to zero, and for the others, use the appropriate fixed values for their respective hidden nodes in the computation of the network's output. Since most of the wavelet coefficients quantize to zero, this gives a significant additional computational savings. We discuss these results, and the impact on compression gain, in the next Section.

4.8 Compression Results

Table 4 shows the percent compression gained over the whole test set, for nets trained with various numbers of hidden units, with and without the speedup algorithm described in the last Section. In these tests, prediction is being done on subbands LH0, HL0, HH0, LH1, and HL1. We did not use prediction on subband HH1 since, as shown above, very little learning is achieved on that subband[4]. In order to compute the compression gain, the system was first run on a given test image with prediction switched on. Then, the system was run on the same image, with prediction switched off, in a 'binary search' mode, until the PSNR was equal to that found with prediction switched on; the number of bits required to encode/decode the luma were then compared. The Table also shows the number of multiply-adds per pixel required by the prediction, averaged over the test set. Here the nets were all trained with learning rate $\epsilon = 0.01$[5].

Interestingly, reducing the calls to the network actually improves performance slightly. The problem being solved here is very noisy, and applying the speedup trick amounts to adding prior knowledge to the predictor, namely, that zeros tend to cluster. These results lead us to ask whether further speedup gains can be achieved by relaxing the context test (i.e. reducing the number of coefficients, at least one of whose values must quantize to a nonzero value, in order to call the net). We performed

[4]Tests with HH1 added gave almost identical compression results.

[5]Here the ping-pong ordering in the PWC codec was switched off: this consistently gave slightly better results.

	Compression Gain (%)		Operations Per Pixel	
# Hidden Units	Speedup	No Speedup	Speedup	No Speedup
5	4.3	4.1	30	109
10	4.7	4.4	61	219
20	4.9	4.7	121	438
30	5.0	4.8	182	656
40	4.9	4.7	243	875

Table 4: Compression results, with and without the 4-coefficient context-text speedup.

a sensitivity analysis, using the first training image, to determine which coefficients in the context have the most influence on prediction accuracy. Labeling the coefficient to be predicted q, this showed that the two most important coefficients in the context were the one to the immediate left of q (p_1) and the coefficient immediately above q (p_2). Table 5 shows the result of calling the net only if p_1 or p_2 quantizes to a nonzero value. For a slight reduction in compression gain, computational complexity is again reduced, by about 25%:

	Compression Gain (%)		Operations Per Pixel	
# Hidden Units	Speedup	No Speedup	Speedup	No Speedup
5	4.3	4.1	22	109
10	4.6	4.4	45	219
20	4.9	4.7	89	438
30	5.0	4.8	134	656
40	4.8	4.7	179	875

Table 5: Compression results, with the 2-coefficient context-test.

Finally, we implemented the sparseness speedup described in the previous Section. Since the predictor will usually predict a non-zero value, we first set to zero any reconstructed coefficient which, when quantized, would become zero (this introduces between one and two operations per pixel, for those subbands for which prediction is done). This has the added benefit of further reducing the number of calls to the network. We also combined this with the 2-coefficient context test described above. We estimated the overhead of the sparse forward propagation as 24 (size of context) operations per call to the network . The results of this experiment are shown in Table 6.

Thus by combining speedup methods, we were able to reduce the number of operations by a factor of approximately 10, with no loss in compression gain.

The variation in compression gain between images is shown in Table 7, for the 30-hidden units network with the 4-coefficient context test algorithm, again on the test images.

	Compression Gain (%)		Operations Per Pixel	
# Hidden Units	Speedup	No Speedup	Speedup	No Speedup
5	4.3	4.1	14	109
10	4.5	4.4	25	219
20	4.8	4.7	45	438
30	4.9	4.8	65	656
40	4.7	4.7	85	875

Table 6: Compression results, with the 2-coefficient context-test combined with the sparse forward propagation algorithm.

Test Image	Compression Gain (%)
18	4.3 (4.2)
19	6.5 (6.5)
20	4.3 (4.4)
21	5.0 (4.8)
22	5.4 (5.2)
23	6.1 (5.9)
24	4.6 (4.5)

Table 7: Compression results for 30-hidden unit net, with 4-pixel context test speedup algorithm, and corresponding 2-pixel test in parentheses. Mean (macro-averaged) gain per image = 5.0%

5 Discussion

What kind of nonlinear prediction are these nets performing? The images for which the compression gain (Table 7) was highest had many sharp edges (a picket fence). However, the presence of straight edges is not necessary: image 23 is of two parrots. Predicting the coefficients along edges requires nonlinear computations, and clearly the gain is partly from that. Figure 10 shows the residuals, with and without prediction, for one of the images in the training set, for a subset of the LH0 subband in which vertical lines were prevalent in the original image. The corresponding image segment is also shown. The nonlinear prediction is able to model the edges well, thus reducing the variance of the residuals. We also verified this hypothesis directly, by creating a synthetic 512 by 512 image consisting of a 1-pixel wide vertical line placed every third pixel in the image. On this image, adding prediction gave 17% better compression at the same PSNR. This kind of coefficient-prediction can be seen as a generalization of [10], where compression gains were demonstrated by predicting the signs of wavelet coefficients. Finally, note that the biggest compression gain occurs in subbands LH0, HL0 and HH0: there the gain, averaged over test images (with the 4-coefficient context test) is approximately 10% on each, compared with between 1%

and 2% on LH1 and HL1[6].

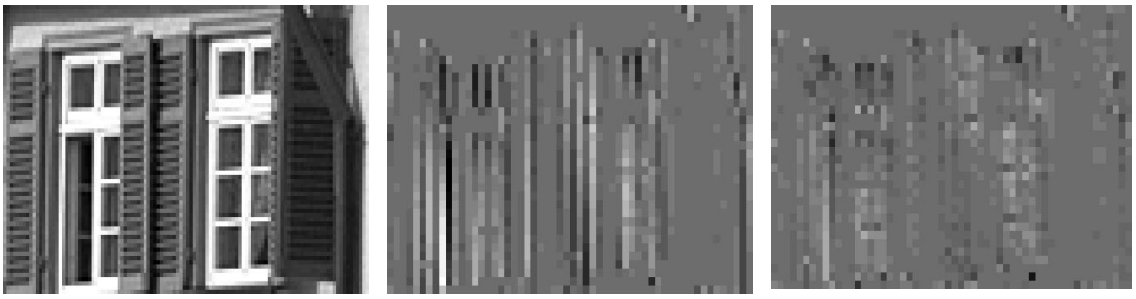

Figure 10: Example of prediction efficiency. Left: image section under test. Center: wavelet coefficients before prediction (amplitudes plotted as pixel intensities; zero is mid gray). Right: after prediction – note how the network is efficient in predicting edges.

6 Conclusions

We have shown that the addition of a nonlinear prediction step using a neural network yields per-image compression gains of between 4.3% and 6.5%, and that a 4% gain can be achieved with approximately 15 extra operations per pixel. How might we use prediction elsewhere in the compression algorithm to further improve this result? As in [11], we can expect that predicting the range of wavelet coefficients can also yield additional compression. This is because the quantization step size can be adapted to each coefficient, according to its estimated prange. As with $F(\tilde{C}_{i<}, W)$, we could predict the ranges by using another nonlinear predictor $S(\tilde{C}_{i<}, W)$. Finally, although neural network predictors can be slow, we have shown that the loss in speed can be largely corrected by judicious use of how and when the net is used.

7 Acknowledgements

We wish to thank John Platt, of Microsoft Research, for useful discussions.

References

[1] S. Mallat. *A wavelet tour of signal processing.* academic press, 1998.

[2] H. S. Malvar. Fast progressive wavelet coding. In *Proceedings of IEEE Data Compression*, pages 336–343, 1999.

[3] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. on Signal Processing*, 41:3445–3462, 1993.

[6] Approximately 46% of the bits used to encode the images are used on the LH0, HL0 and HH0 subbands, again averaged over the 7 test images.

[4] A. Said and W. A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. Circuits and Systems for Video Technology*, 6:243–250, 1996.

[5] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. *Advances in Neural Information Processing Systems*, 9:155–161, 1997.

[6] P.J. Green and B.W. Silverman. *Nonparametric Regression and Generalized Linear Models*. Chapman and Hall, 1994.

[7] Y. LeCun, J.S. Denker, S. Solla, R.E. Howard, and L.D. Jackel. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, pages 598–605. Morgan Kaufman, 1990.

[8] J. Liu and P. Moulin. Analysis of interscale and intrascale dependencies between image wavelet coefficients. In *Proc. IEEE Int. Conf. on Image Processing*, Vancouver, Canada, Sept. 2000.

[9] Y. LeCun and Y. Bengio. Convolutional networks for images, speech and time-series. In M. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1995.

[10] A. Deever and S.S. Hemami. What's your sign?: efficient sign coding for embedded wavelet image coding. In *Proc. IEEE Data Compression Conf.*, pages 273–282, April 2000.

[11] Scott M. LoPresto and Kannan Ramchandran. Image coding based on mixture modeling of wavelet coefficients and a fast estimation-quantization framework. In *Data Compression Conference, Snowbird, Utah*, 1997.