

IMAGE COMPRESSION WITH ON-LINE AND OFF-LINE LEARNING

Patrice Y. Simard, Christopher J.C. Burges, David Steinkraus, Henrique S. Malvar

Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
{patrice, cburges, v-davste, malvar}@microsoft.com

ABSTRACT

Images typically contain smooth regions, which are easily compressed by linear transforms, and high activity regions (edges, textures), which are harder to compress. To compress the first kind, we use a “zero” encoder that has infinite context, very low capacity, and which adapts very quickly to the content. For the second, we use an “interpolation” encoder, based on neural networks, which has high capacity, a finite-size context, and is trained off-line. The two encoders can be used separately or in combination. The zero-encoder surpasses JPEG2000 by 3.5% in overall compression, even though it is less efficient in high activity regions. Thanks to off-line training, the interpolation-encoder predicts high activity regions well, so it also matches the performance of JPEG2000, even though it does not use an arithmetic encoder and is less efficient in low activity regions. In both cases it is surprising that we match the state-of-the-art in image compression without using adaptive arithmetic encoding.

1. INTRODUCTION

Most wavelet-based signal compression systems [1, 2] are based on the structure shown in Figure 1. A linear transform, e.g. a wavelet transform, is applied to the image, followed by a quantization step. In this step, the wavelet coefficients are quantized by dividing them by a quantization factor Q and then rounding them to nearest integers. Higher amounts of compression are obtained by increasing the quantization factor Q . The resulting integers are then split into bitplanes, and each bitplane is encoded without loss using a context table and an entropy coder. When a bitplane is encoded, a probability for ‘0’ or ‘1’ is computed

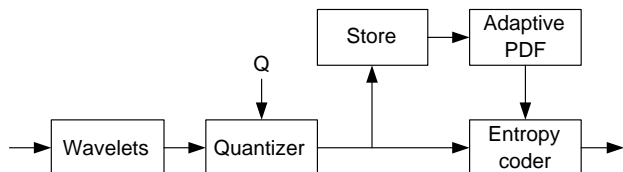


Fig. 1. Traditional compression system

for each bit by looking at previously encoded bits at adjacent locations. The previously encoded bits are called “context”; a typical context shape is shown Figure 2. From the context, a probability distribution function (or PDF) can be computed for the next bit to be coded (depicted by “?” in the figure). Given the bit to be coded and the PDF, the entropy coder can encode each bit in a near optimal fashion (within 0.5% of true entropy). The PDF is adapted by keeping track, for each context value, of how many ‘0’s and ‘1’s were encoded. If the context has N bits, the PDF context table has 2^N entries.

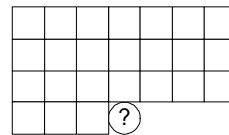


Fig. 2. A typical context region shape

From a learning perspective, these tables can be problematic. If the context is too large (e.g. 18 bits), the context table has too many degrees of freedom, and there will not be enough data to train the PDF tables within a single image. Conversely, if the context is too small, the accuracy of the estimation of the PDF will be degraded.

To circumvent the problem, we propose two algorithms which do not use context tables. Our algorithms are based on the assumption that typical images are made of smooth regions of low activity and regions of high activity, generally resulting from edges or textures. The first algorithm has low learning capacity, fast on-line adaptation, and an infinite-size context. It is best suited for regions of low activity, which are essentially composed of zeros. It is described in Section 2. In contrast the second algorithm has a fixed-size context, a controlled capacity (independent of the context size), and is adapted off-line. The size of the context is determined by the capacity needed for learning and by the desired computational speed. This algorithm works best in regions of high activity. It is described in Section 3.

2. ON-LINE LEARNING: THE TARP FILTER

In this section we review the encoder based on a Tarp filter [3]. The basic idea behind the Tarp filter is to produce a simple but efficient recursive density estimator. For 1-D signals, a simple recursive PDF predictor has the form

$$p[t] = ap[t - 1] + (1 - a)v[t] \quad (1)$$

where $p[t]$ is the estimate of the probability of getting a one for position $t + 1$, $v[t]$ is the observed value (0 or 1) at position t , and $0 < a < 1$ is a learning rate (speed of adaptation) parameter; the closer a is to 1, the slower the learning rate. We see that p is the convolution of $v[t]$ with the filter impulse response $f[t] = a^t(1 - a)$ for $t > 0$, and 0 otherwise. f itself is a PDF, and since $v[t]$ is either 0 or 1, it follows that $p[t]$ is a sum of Parzen windows [4], shaped by f .

In [3] we extend the filter in (1) to 2-D, via a top-to-bottom, left-to-right processing

$$\begin{aligned} p[i, j] &= a(p_1[i, j - 1] + p_2[i - 1, j]) \\ p_1[i, j] &= ap_1[i, j - 1] + \frac{(1-a)^2}{2a}v[i, j] \\ p_2[i, j] &= p_1[i, j] + ap_2[i - 1, j] \end{aligned} \quad (2)$$

followed by a right-to-left processing

$$\begin{aligned} p_2[i, j] &= p_2[i, j] + ap_3[i, j + 1] \\ p_3[i, j] &= ap_3[i, j + 1] + \frac{(1-a)^2}{2a}v[i, j] \end{aligned} \quad (3)$$

As we show in [3], the 2-D estimators above correspond to filtering the probability estimates through a tarp-shaped filter, and hence the name.

The Tarp filter can be used in a wavelet-based image encoder based on bit-plane coding (e.g. [6]). We can encode (and decode) each bit $v[i, j]$ in a bit plane using an arithmetic coder (AC) that uses the probability estimate $p[i, j]$ for that bit. That is, we use the probability estimates from the Tarp filter to replace the usual probability estimates derived from context tables.

2.1. Experimental Results

We modified the PWC codec in [6] to use the Tarp filter. For each wavelet subband (LH, HL, HH), the Tarp filter is run independently on each bit plane. The probability estimate generated by each bit-plane Tarp is then used by the arithmetic encoder to encode each bit-plane symbol. For each non-zero symbol, the sign is appended to the bitstream without compression. Additional compression gains could be achieved by compressing the signs [7].

A typical set of results for the image set is shown in Table 1, for a peak signal-to-noise ratio (PSNR) of 40.0 dB (high visual quality). Similar results are obtained at other

Table 1. Compression results; Kodak set, PSNR = 40 dB

	JPEG	EZ-HLBT [9]	JPEG2000 [1]	Tarp [3, 8]
bits/pixel	1.62	1.42	1.39	1.34
% diff	-16.5%	-2%	0	+3.5%

PSNR settings. For comparison with embedded zerotree-based coders, we have also included results for the EZ-HLBT (embedded zerotree with hierarchical lapped biorthogonal transform) codec [9]. It is surprising that the Tarp-based codec performs better than JPEG2000 on the Kodak set, since JPEG2000 uses more sophisticated context modelling, and is thus much more complex than Tarp. We note that the Tarp codec is not using contextual information other than the distance of non-zero bits with respect to the currently coded bit. So, the Tarp filter is probably sub-optimal in coding regions with edges, which we address in the next section.

3. OFF-LINE LEARNING: NONLINEAR PREDICTOR

In Section 2, the second half of Figure 1 was replaced with a simple, low capacity adaptive filter; this gave excellent results. Results are now given on using a high capacity learning machine, trained off-line, to replace the quantizer box [10]. The new quantizer is shown in Figure 3. There,

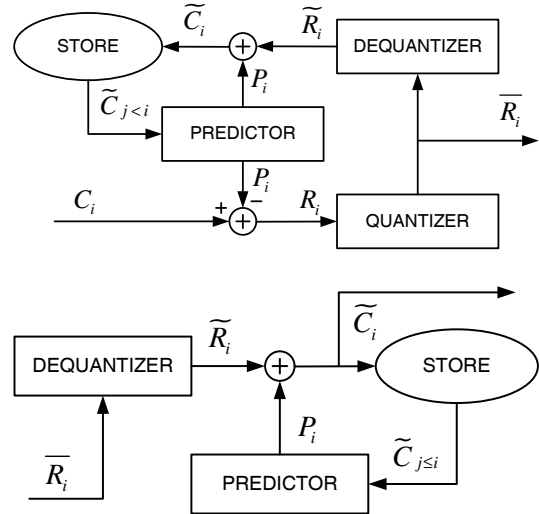


Fig. 3. New quantizer with a nonlinear predictor: During the encoding process (top), for each coefficient C_i , the predictor box outputs a prediction P_i . The residual R_i from the prediction is then quantized as \bar{R}_i . The predictor uses a context of previously encoded coefficients, which is also available at the decoder (bottom).

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 \tilde{R}_i and \hat{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 (Figure 3, bottom), 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.

To increase compression, we must decrease the entropy of the residual coefficients R_i . If we start with the simplest predictor $P_i = 0$, the residual and wavelet coefficients are identical. The wavelet coefficients of 2D images usually have a histogram that is well approximated by the Laplacian distribution $\frac{a}{2} \exp^{-a|x|}$ (assuming zero mean). It is straightforward to show that the entropy (assuming fixed, small bin size) varies approximately as the log of the variance. Thus, decreasing the variance for a fixed bin width is here equivalent to decreasing the entropy. This argument assumes that the process of decreasing the variance does not significantly change the distribution itself; for example, since the distribution with maximum entropy for given mean and variance is a Gaussian, the entropy could increase despite its variance decreasing if the distribution becomes more Gaussian.

We can now concentrate on the problem of minimizing the variance $\sum_i R_i^2$, which unlike the entropy can be computed locally by minimizing each R_i^2 . 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 [11] or generalized linear models [12]. We chose to use a neural network since its computational cost can be directly controlled and training is straightforward.

3.1. Training

Several different architectures were investigated; here, we report results using a network with 24 inputs, 30 hidden units and one output unit. For details on comparisons between different network architectures and training methodologies, see [10]. The inputs correspond to the causal context shown in Figure 2. For each subband and wavelet level, the inputs are rescaled so that 7 standard deviations lie in $[-1, 1]$, and network outputs are also inversely rescaled. The output is sigmoidal; a linear output unit gave much worse performance, due to the long-tailed distribution of the data.

The first 16 images in the Kodak database were used for training, image 17 for validation, and the next 7 images as a test set. The amount of training data for LH0 is 1.6 million patterns, and drops by a factor of 4 going up each wavelet

level. The variance of the data also rapidly increases with wavelet level. It is therefore not surprising that the best performance was found for the LH0 net. The net used for HL coefficients was the same as that used for the corresponding LH coefficients, but taking the transpose of the coefficients matrix first.

Training off-line, on static data, revealed an interesting problem at higher wavelet levels. Because the net is used in a loop, the system is actually dynamic, and feedback was found to cause oscillations in reconstructed images. This problem was solved by training the nets themselves 'in loop' (see Figure 3), that is, using contexts of reconstructed wavelet coefficients. Any such oscillations are then penalized as errors. The training is still off-line, in that a fixed training set is used, but the training data seen by the network is dynamically generated. This was found to reduce the error due to oscillations alone by over a factor of 100. However this has the drawback that a given quantization level must be chosen for training, as opposed to training on static wavelet coefficients. We chose $Q = 10$ since the resulting distortion is very small. Training was stopped after no further reduction in mean squared error on the validation image was observed.

3.2. Results

Using a forward prop for each coefficient comes at considerable computational cost: overall, the mean number of multiply adds per pixel to incorporate prediction is 656. To solve this problem, a sensitivity analysis of the causal context was done, to find those coefficients that were most important. The four most important coefficients, which were found to be those contiguous to the coefficient to be predicted (see Figure 2), were then used in a test: if all those coefficients become zero after quantization, predict 0 (i.e. do not call the net). This reduced the number of multiply adds per pixel due to prediction down to 182, and it actually improved overall compression performance slightly. Another speedup method, which takes advantage of the sparseness of the net's inputs, gave further computational savings. Table 2 shows the results on the test set, micro-averaged over all test images. Prediction was used only for levels LH0, HLO,



Fig. 4. Example of prediction efficiency on edges. The center image are the LH0 wavelet coefficients before prediction (amplitudes plotted as pixel intensities; zero is mid gray). The right image is after prediction.

Table 2. Compression results using prediction with 4-coefficient speedup.

Test Image	Compression Gain (%)	Gain Relative to JPEG2000 (%)
18-24	5.3	-0.6

HH0, LH1, and HL1, since the other bands gave very little gain. Comparisons are made at fixed PSNR (found for the non-prediction results by using a binary search, to match that found with prediction). Also shown is the corresponding result for JPEG2000. The best improvement was for an image with many edges; JPEG2000 did better for smoother images. Figure 4 shows the wavelet coefficient before and after prediction for a piece of a test image with many edges.

The nonlinear predictor removes the edges without amplifying the noise and gains over JPEG2000 in regions with high activity. However the Golomb-Rice encoder that we used to encode the residual cannot compete with JPEG2000's arithmetic encoder and this algorithm loses to JPEG2000 in regions of low activity.

4. CONCLUSION

Virtually all state-of-the-art compression algorithms use arithmetic encoding, combined with an adaptive context table. From a learning perspective these approaches leave a lot to be desired, because they do not provide capacity control. We have proposed two radically different solutions to this problem. The first algorithm, the Tarp filter, has an infinite size context and low learning capacity. Despite the fact that it completely ignores directional information, it yields better compression performance than JPEG2000. The reason, we believe, is that in regions of low activity, the Tarp filter better adapts its estimate of the PDF than JPEG2000.

The second algorithm replaces the adaptive table with a learning machine of controlled capacity. A neural network is used as a predictor for wavelet coefficients, with a fixed size context and off-line training. The residuals are then encoded with a Golomb-Rice adaptive encoder. Despite the simplicity of this adaptive encoder, the neural network based encoder matches JPEG2000 performance. Contrary to the tarp filter, we believe this encoder performs poorly in low activity regions, but makes it up in regions with edges and textures.

Image compression is a well-studied field, in which it is difficult to achieve significant improvements. Interestingly, capacity control was an overlooked aspect, and we were able to improve or match the state of the art with simple learning algorithms. This opens the door to new and interesting research avenues, such as improving the tarp fil-

ter, trying new nonlinear predictors (SVMs, decision trees) and combining the two approaches.

REFERENCES

- [1] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards, and Practice*. Boston, MA: Kluwer, 2002.
- [2] P. Y. Simard and H. S. Malvar, "A wavelet coder for masked images," *Proc. IEEE Data Compression Conf.*, Snowbird, UT, pp.93–102, Mar. 2001.
- [3] P. Y. Simard, D. Steinkrauss, and H. S. Malvar, "Online adaptation in image coding with a 2-D tarp filter," *Proc. IEEE Data Compression Conf.*, Snowbird, UT, pp.23–32, Apr. 2002.
- [4] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. New York, NY: Wiley and Sons, 1973.
- [5] R. J. Stevens, A. F. Lehar, and F. H. Preston, "Manipulation and presentation of multidimensional image data using the Peano scan," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 5, pp. 520–533, Sept. 1983.
- [6] H. S. Malvar, "Fast progressive wavelet coding," *Proc. IEEE Data Compression Conf.*, Snowbird, UT, pp. 336-343, Mar. 1999.
- [7] A. Deever and S. S. Hemami, "What's your sign?: efficient sign coding for embedded wavelet image coding," *Proc. IEEE Data Compression Conf.*, Snowbird, UT, pp. 273–282, Mar. 2000.
- [8] J. E. Fowler, "QccPack: an open-source software library for quantization, compression, and coding," see <http://qccpack.sourceforge.net/main.html>.
- [9] H. S. Malvar, "Lapped biorthogonal transforms for transform coding with reduced blocking and ringing artifacts," *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, Munich, Germany, pp. 2421–2424, April 1997.
- [10] C. J. C. Burges, P. Simard, and H. S. Malvar, "Improving wavelet image compression with neural networks," *Microsoft Research Tech. Rep.* MSR-TR-2001-47, June, 2001.
- [11] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, and V. Vapnik. "Support Vector Regression Machines," in *Advances in Neural Information Processing Systems*, 9:155–161, 1997.
- [12] P.J. Green and B.W. Silverman. *Nonparametric Regression and Generalized Linear Models*. Chapman and Hall, 1994.