Image Compression with Competitive Networks and Pre-fixed Prototypes*

Enrique Mérida-Casermeiro[1], Domingo López-Rodríguez[1], and Juan M. Ortiz-de-Lazcano-Lobato[2]

[1] Department of Applied Mathematics, University of Málaga, Málaga, Spain; {merida,dlopez}@ctima.uma.es

[2] Department of Computer Science and Artificial Intelligence, University of Málaga, Málaga, Spain; jmortiz@lcc.uma.es

Abstract. Image compression techniques have required much attention from the neural networks community for the last years. In this work we intend to develop a new algorithm to perform image compression based on adding some pre-fixed prototypes to those obtained by a competitive neural network. Prototypes are selected to get a better representation of the compressed image, improving the computational time needed to encode the image and decreasing the code-book storage necessities of the standard approach. This new method has been tested with some well-known images and results proved that our proposal outperforms classical methods in terms of maximizing peak-signal-to-noise-ratio values.

1 Introduction

The storage or transmission of images are tasks demanding either large capacity and/or bandwidth. Thus, image compression is key in the development of various multimedia computer services and telecommunication applications, such as medical imaging [?], satellite transmission, teleconferencing, pattern recognition [?], etc. The goal of image coding is to reduce both the distortion introduced in the coding process and the bit rate (or, equivalently, the compression rate) to an acceptable level.

Recent publications show a substantial increase in the use of neural networks for image compression and coding. For a review in neural techniques for image compression, refer to [?].

Vector Quantization (VQ) is a lossy compression technique that can achieve high compression rates with good visual fidelity, see for example [?]. VQ is a coding method designed to represent a multidimensional space by means of a finite number of vectors, called representatives, prototypes or code-vectors. A vector quantizer statistically encodes data vectors in order to quantize and compress the data, by mapping each input vector in the p-dimensional Euclidean space $\mathbb{R}^p$ into one of the K prototypes.

* This work has been partially supported by Junta de Andalucía project number P06-TIC-01615.

According to Shannon's rate distortion theory, VQ can always achieve better compression performance than any conventional coding technique based on the encoding of scalar quantities [?].

Linde, Buzo and Gray [?] proposed the well-known LBG algorithm for VQ which made no use of differentiation, and it is the standard approach to compute the codebook (the set of prototypes).

Competitive networks are designed to cluster the input data. Thus, by using VQ techniques in this type of networks, tasks such as data coding and compression can be performed.

The basic structure of a network of this type is as follows: given an input vector from a p-dimensional space, K neurons compute the VQ code-book in which each neuron relates to one code-vector via its coupling weights. The weight $\mathbf{w}_i = (w_{i,1}, \ldots, w_{i,p})$ associated with the i-th neuron is eventually trained to represent the code-vector $\mathbf{c}_i$ in the code-book. As the network is being trained, all the weights will be optimized to represent the best possible partition of all the input vectors.

The aim of this paper is to present a new technique for image compression based on competitive neural networks. This technique allows to reduce the number of code-vectors needed to achieve a high quality code-book. This method is based on the use of a combination of multiple prototypes to store one input vector, achieving a better representation of the input dataset.

2 The Standard Approach

Let us consider an image $I(i, j)$. In order to compress the image by means of VQ competitive networks, the image is subdivided into $N \times M$ square sub-images of size $k \times k$, called *windows*. These windows are our input patterns with $p = k^2$ components (these patterns are obtained by arranging the pixel values row by row from top to bottom).

The compression process consists in selecting a reduced set of K representative windows (corresponding to the solution prototypes) and replacing each window of the original image with the closest representative window among the prototypes. Thus, the compressed image I' is built.

In this work we have used the standard competitive learning (SCL) rule to build the code-book. Let $X = \{x_1, \ldots, x_n\}$ be the set of input patterns, and let $\{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$ represent the code-book. The algorithm is as follows:

1. Choose an input pattern, x_i.
2. Compute the winning prototype (the closest to x_i), $\mathbf{w}_c$, verifying
$$\|x_i - \mathbf{w}_c\|^2 = \min_{j=1,\ldots,K} \|x_i - \mathbf{w}_j\|^2$$
3. Update the vector $\mathbf{w}_c$ according to
$$\mathbf{w}_c(t+1) = \mathbf{w}_c(t) + \alpha(t) \cdot (x_i - \mathbf{w}_c(t))$$

where $\alpha(t)$ is the learning rate parameter, usually convergent to 0 as the number of iterations t tends to infinity. A typical value for $\alpha(t)$ is given by a linear decrease from 0.9 to 0 along the iterations.

4. Increase t, and repeat steps 1-4 until convergence is detected.

The bottleneck of this algorithm is the computation of the closest prototype, since it involves the calculation of K distances. Thus, the time spent by this algorithm to build the code-book when the whole set of input patterns is presented to the net is proportional to $n \cdot K$.

Though many other learning rules have been developed to obtain quasi-optimal code-books [**?**, **?**, **?**, **?**, **?**], we have used the SCL algorithm since its performance is very well-known and allows us to compare our technique to the standard approach.

3 Pre-fixed Prototypes

Our proposal differs from the standard approach in several points:

- Some of the prototypes used in the compression are *a priori* known by the encoder (transmitting the image) and decoder (receiving the image). Thus, with some prototypes pre-fixed in advance, the size of the code-book is reduced considerably.
  So, if, for example, we have $K = 32$ prototypes, and we fix $K' = 16$ of them, then the encoder only has to transmit $K - K' = 16$ prototypes after the encoding phase.
- In addition, since not all prototypes need to be computed, the encoding algorithm (the competitive neural network) will spend less time in computing the code-book.

Thus, with our proposal, we achieve a high quality compression by reducing computational time as well as the code-book storage space. Our technique maintains the same compression rate as the standard approach for the given number of prototypes.

This means that the transmission of the image from the encoder to the decoder (receiver) is improved, since the codebook storage space is halved.

The encoding algorithm is based in applying the standard competitive learning rule (SCL) mentioned above, but with the restriction of the pre-fixed prototypes:

1. Choose an input pattern, x_i.
2. Compute the winning prototype (the closest to x_i), $\mathbf{w}_c$, verifying
$$\|x_i - \mathbf{w}_c\|^2 = \min_{j=1,\ldots,K} \|x_i - \mathbf{w}_j\|^2$$
3. If the winning unit $\mathbf{w}_c$ represents one of the fixed prototypes, it does not learn. Otherwise, update the vector $\mathbf{w}_c$ according to
$$\mathbf{w}_c(t+1) = \mathbf{w}_c(t) + \alpha(t) \cdot (x_i - \mathbf{w}_c(t))$$
where $\alpha(t)$ is the learning rate parameter.

4. Increase t, and repeat steps 1-4 until convergence is detected.

It seems reasonable to use $K' = \frac{K}{2}$, that is, to fix half the prototypes, since it allows an easy codification of each input pattern: if b bits $m_0, \ldots, m_{b-1}$ are used to encode a pattern, then m_0 will indicate whether the corresponding prototype is *a priori* known by both encoder and decoder ($m_0 = 0$) or not ($m_0 = 1$). The other $b-1$ bits are used to encode the prototype number, from 0 to K'. So, the decoder, when a sequence of bits arrives, studies m_0 and looks for the prototype in the corresponding part of the code-book (the fixed or the transmitted, depending on m_0).

The intuitive idea behind the utilization of pre-fixed prototypes is that these prototypes are designed to fit in areas of the image in which there is little detail (sky, rivers, etc.), whilst the learnt prototypes focus on the areas of great detail.

4 Experimental Results

In this section we compare the efficiency of our proposal in image compression with respect to the standard approach.

We have considered a test set of images formed by 4 images (the well-known *Lenna* image plus 3 images from MatLab Image Processing Toolbox, see Fig. 1). Each image in our experiments has 256×256 pixels, and window size is 4×4. So, the number of input patterns for the standard algorithm is 4096. The number K of representative windows varies in the set $\mathcal{K} = \{32, 64\}$.

Two measures has been used in this work to compare the efficiency of our technique

– The *peak-signal-to-noise-ratio* (PSNR), defined as:
$$\text{PSNR} = 20 \log_{10} \left(\frac{255}{\text{RMSE}} \right)$$
and measured in decibels (dB), where the image has 256 gray levels and RMSE is the root mean square error between two images (the original and the compressed). Ideally, a value of PSNR $= \infty$ is the goal to attain, since it corresponds to a lossless compression. Thus, one tries to obtain the maximum value possible for PSNR.
– The 1-norm ($\|\cdot\|_1$) of the difference between the original image I and the compressed image I', defined as:
$$\|I - I'\|_1 = \max_j \sum_i |I(i,j) - I'(i,j)|$$
where $I(i,j)$ represents the value of the pixel (i,j) in the image I.
A lower value of this norm indicates a better approximation of the compressed image to the original one.

Some of the test images are shown in Fig. 1. The compressed images are shown in Figs. 2 and 3. In these figures, we can observe that our compressed images obtain higher visual fidelity than the standard approach.

Fig. 1. Sample images from the test set: lenna (top left), cameraman (top right), rice (bottom left) and kids (bottom right).

For our proposal, $\frac{K}{2}$ fixed prototypes were built by considering vectors of the form $\mathbf{w}_i = (c_i, \dots, c_i) \in \{0, \dots, 255\}^{k^2}$, that is, all components are equal, each prototype representing a window with a constant grey-level in all its pixels. c_i values ($i = 1, \dots, \frac{K}{2}$) were equally spaced in $\{0, \dots, 255\}$. The same pre-fixed prototypes were used for every test image.

In Table 1, there are represented the PSNR values of the compressed images using both the standard and our approach. It can be noted that if our method is used with $K' = K/2$ fixed prototypes, where K is the number of the standard approach, the PSNR value obtained is at least comparable, and better in some cases, to that of the standard approach.

This surprising fact may be explained as follows: since $K' = \frac{K}{2}$ prototypes are pre-fixed, the dimensionality of the problem is reduced to half. As the dimension is reduced, the number of possible local minima of the distortion function is also reduced. So, the learning phase can avoid certain bad local minima present when all K prototypes are considered.

This is also possible since prototypes with a constant gray-level in all its components are usually present in wide regions of most images (regions with little detail).

I can also be observed that the 1-norm of our proposal is lower in most cases than the corresponding of the standard approach.

Table 1. PSNR and $\|\cdot\|_1$ results for the considered compressed images. The quotient Δt is defined by $\Delta t = \dfrac{t_{\mathrm{new}}}{t_{\mathrm{old}}}$.

		Standard Approach			Prop.			
Image	K	PSNR	$\|\cdot\|_1$	t_{old}	PSNR	$\|\cdot\|_1$	t_{new}	Δt
cameraman	32	23.94	23.6	33.65	24.13	20.1	31.38	0.93
rice	32	28.00	11.8	33.31	28.47	10.5	32.01	0.96
lenna	32	25.41	16.9	33.46	25.39	16.0	31.12	0.93
kids	32	27.18	12.1	33.56	26.79	13.8	31.17	0.93
cameraman	64	25.32	19.3	53.40	25.16	18.9	45.64	0.85
rice	64	29.41	8.6	45.54	30.33	7.6	43.50	0.95
lenna	64	26.73	13.6	46.31	26.48	15.7	43.00	0.93
kids	64	27.92	11.8	53.73	27.87	11.24	42.18	0.78

The time spent by our algorithm to build the code-book is also lower than the standard, as shown in the last column of Table 1, where Δt indicates the quotient between the time spent by the standard approach and the time used by our proposal.

5 Conclusions

In this work we have presented a new algorithm to perform image compression based on combining fixed prototypes with the solution prototypes obtained by a competitive learning rule.

This new method is able to represent compressed images with higher visual fidelity than the standard approach, at the same time that achieves greater PSNR values in many cases. In addition, with this method, the computation of the code-book is less time-consuming, since fixed prototypes are never updated.

We have tested our approach using the standard competitive learning rule. Better results are expected if learning rules as [?, ?, ?] are used instead.

Our future work covers the study of new algorithms taking advantage of combining fixed and variable prototypes to form different windows in the compressed image. The development of a competitive learning rule to optimize the code-book is also an issue of research.

The application of this technique to specific scenarios or groups of images (biomedical images, satellite images, etc.) can improve their processing, segmentation, compression and transmission.

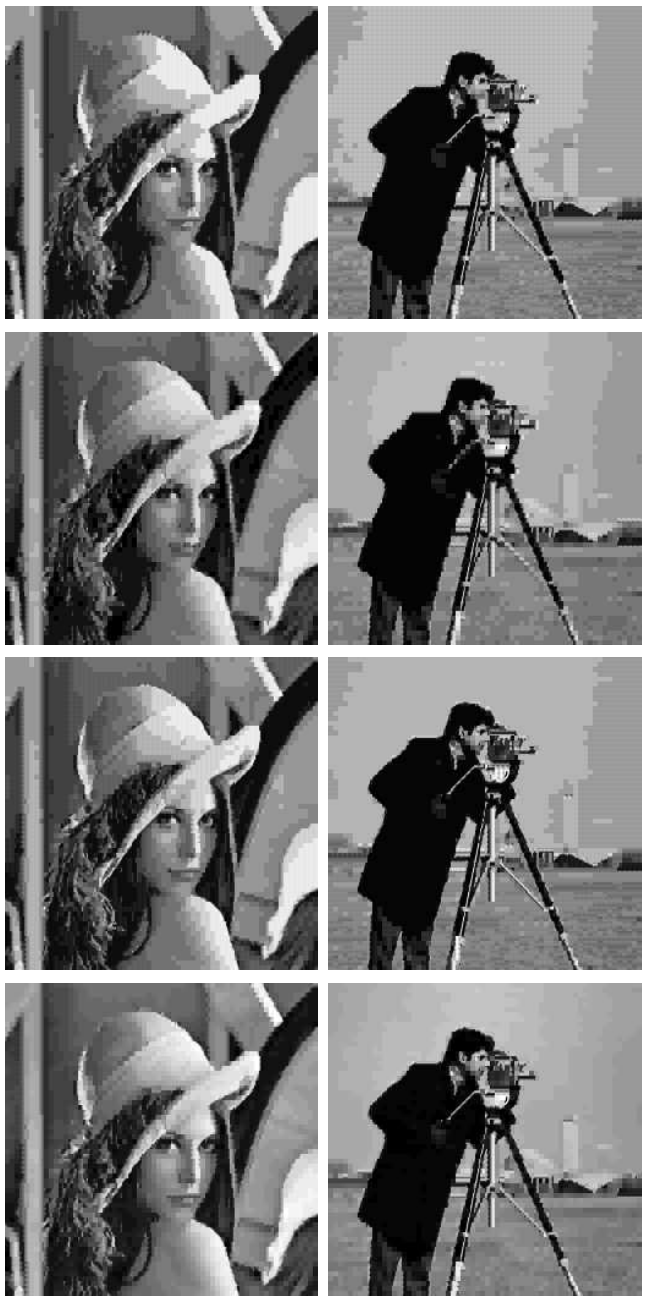

Fig. 2. Compressed images (from top to bottom): standard approach with 32 prototypes, our proposal with 32 prototypes, standard approach with 64 prototypes and our proposal with 64 prototypes.

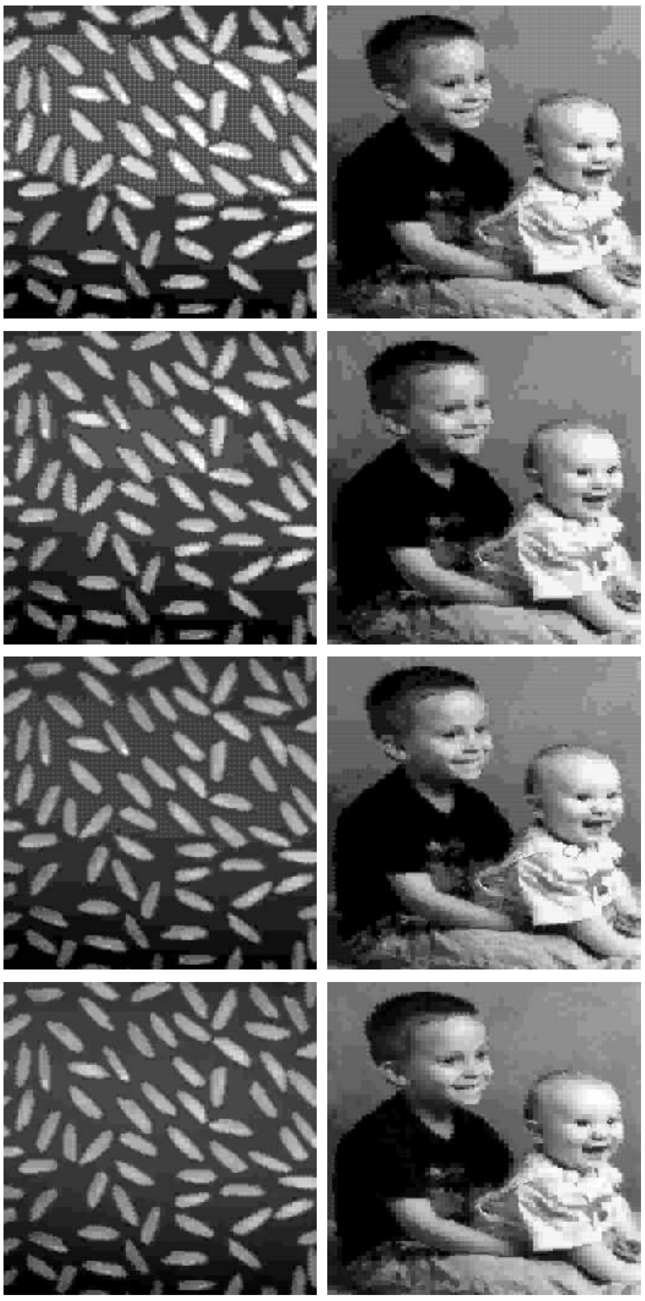

Fig. 3. Compressed images (from top to bottom): standard approach with 32 prototypes, our proposal with 32 prototypes, standard approach with 64 prototypes and our proposal with 64 prototypes.