Available online at www.sciencedirect.com
SCIENCE DIRECT®
Applied Soft Computing 6 (2006) 258–271
Applied Soft Computing
www.elsevier.com/locate/asoc

A neural networks approach to image data compression☆

Hamdy S. Soliman, Mohammed Omari *

Computer Science Department, New Mexico Institute of Mining and Technology, Socorro, NM 87801, USA

Received 2 June 2004; received in revised form 1 November 2004; accepted 13 December 2004

Abstract

We present a novel neural model for image compression called the direct classification (DC) model. The DC is a hybrid between a subset of the self-organizing Kohonen (SOK) model and the adaptive resonance theory (ART) model. The DC is a fast and efficient neural classification engine. The DC training utilizes the accuracy of the winner-takes-all feature of the SOK model and the elasticity/speed of the ART1 model. The DC engine has experimentally achieved much better results than the state-of-the-art peer image compression techniques (e.g., JPEG2000 and DjVu wavelet technology) especially in the domains of colored documents and still satellite images. We include a comprehensive analysis of the most important parameters of our DC system and their effects on system performance.

Keywords: Image compression; Kohonen model; ART model; Direct classification; Self-organizing feature map; Geosynchronous satellite; Colored documents; Universal codebook

1. Introduction

Uncompressed multimedia data require considerable storage capacity and transmission bandwidth. Thus, image compression is a very important factor for better utilization of network bandwidth and computer storage. The compression process is usually lossy and is based on redundancy and irrelevancy reduction, which are inherent in the image domain. Redundancy reduction aims at removing duplication from the signal source (image/video), whereas irrelevancy reduction omits parts of the signal that will not be noticed by the signal receiver.

There are two well-known major approaches to the implementation of image compression: discrete cosine transform (DCT) technique and neural networks models. In the DCT approach, compression is accomplished by applying a linear transform to de-correlate the image data (source encoder), quantizing the resulting transform coefficients (quantizer), and entropy coding the quantized values (entropy encoder). In the neural network approach, a set of input subimage vectors is presented to the net in order to train its neuron's set of weight vectors (weight matrix) via a process of weight adjustment. The neural network approach encompasses the use of the counterpropagation network (CPN) [11],

☆ This work is patented with the US patent office under patent #6608924, issue date: August 19, 2003.

* Corresponding author. Tel.: +1 505 835 5170.
E-mail addresses: hss@nmt.edu (H.S. Soliman), omari@nmt.edu (M. Omari).

1568-4946/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2004.12.006

backpropagation, and the Kohonen self-organizing feature map (KSOFM) [2] models as compression engines.

We introduce a new direct classification (DC) neural model [1] that borrows some features from the adaptive resonance theory (ART) and the KSOFM. Our DC model combines the advantages of these two neural net models in order to achieve high (decompressed) image quality at high compression ratio. The DC model performed exceptionally well in the image document domain, compared to the peer state-of-the-art models (e.g., JPEG2000 and DjVu wavelet technology).

In Section 2, an overview of the DjVu and JPEG2000 peer techniques is presented. We briefly introduce the underlying neural models which are used as the basis of our compression engine in Section 3. Our DC model is defined in Section 4. The metrics used to measure the reconstructed image quality and the compression ratio is shown in Section 5. In Section 6, all major training parameters are discussed and experimentally analyzed. In Sections 7 and 8, we cover some experiments in the domains both of colored documents and satellite images, using the traditional peer models versus our DC model. Section 9 is our conclusion and future work.

2. Wavelet transform techniques

Like the fast Fourier transform (FFT), the discrete wavelet transform (DWT) is a fast, linear operation that operates on a data vector whose length is an integer power of two, transforming it into a numerically different vector of the same length. Also like the FFT, the wavelet transform is invertible and in fact orthogonal—the inverse transform, when viewed as a big matrix, is simply the transpose of the transform [15]. Both FFT and DWT, therefore, can be viewed as a rotation in function space, from the input space (or time) domain, where the basis functions are the unit vectors. For the FFT, this new domain has basis functions that are the familiar sines and cosines. In the wavelet domain, the basis functions are somewhat more complicated; therefore, each wavelet-based image compression scheme has its own transformation and compression functions.

2.1. JPEG2000

Lossy compression removes image details that are (in theory, at least) too small for the human eye to notice. JPEG uses lossy compression [8]. The JPEG compression scheme discards image data to reduce file size.

JPEG compression is based on the fact that our eyes are sensitive to certain kinds of visual details but not others [8]. JPEG analyzes the image and throws out details that it deems unnecessary. How much it throws out is the user choice.

The idea of compressing an image is not new. The discovery of DCT in 1974 [13] is an important achievement for the research community working on image compression. The DCT can be regarded as a discrete-time version of the Fourier cosine series. It is a close relative of DFT, a technique for converting a signal into elementary frequency components. Thus, DCT can be computed with a fast Fourier transform like the algorithm in $O(n \log n)$ operations. Unlike DFT, DCT is real-valued and provides a better approximation of a signal with fewer coefficients. The DCT of a discrete signal $x(n)$, $n = 0, 1, \ldots, N-1$ is defined as:

$$X_t(u) = \sqrt{\frac{2}{N}} C(u) \sum_{n=0}^{N-1} x(n) \cos\left(\frac{(2n+1)un}{N}\right)$$

where $C(u) = 0.707$ for $u = 0$, and 1 otherwise.

In 1992, JPEG established the first international standard for still image compression where the encoders and decoders are DCT-based. The JPEG standard specifies three modes, namely sequential, progressive, and hierarchical for lossy encoding, and one mode of lossless encoding. The ‘‘baseline JPEG coder’’, which is sequential encoding in its simplest form, will be briefly discussed here. Figs. 1 and 2 show the key processing steps in such an encoder and decoder for gray-scale images. Color image compression can be approximately regarded as compression of multiple gray-scale images, which are either compressed entirely one at a time, or are compressed by alternately interleaving 8×8 sample blocks from each in turn.

The DCT-based encoder can be thought of essentially as compression of a stream of 8×8 blocks of image samples. Each 8×8 block makes its

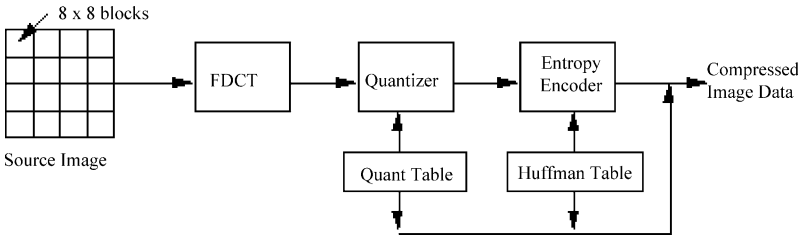


Fig. 1. JPEG encoder block diagram.

way through each processing step and yields output in compressed form into the data stream. Because adjacent image pixels are highly correlated, the ''forward'' DCT (FDCT) processing step lays the foundation for achieving data compression by concentrating most of the signal in the lower spatial frequencies. For a typical 8×8 sample block from a typical source image, most of the spatial frequencies have zero or near-zero amplitude and need not be encoded. In principle, the DCT introduces no loss to the source image samples; it merely transforms them to a domain in which they can be more efficiently encoded.

After output from the FDCT, each of the 64 DCT coefficients is uniformly quantized in conjunction with a carefully designed 64-element quantization table (QT) [8]. At the decoder, the quantized values are multiplied by the corresponding QT elements to recover the original unquantized values. After quantization, all of the quantized coefficients are ordered into the ''zig-zag'' sequence shown in Fig. 3. This ordering helps to facilitate entropy encoding by placing low-frequency non-zero coefficients before high-frequency coefficients. The DC coefficient, which contains a significant fraction of the total image energy, is differentially encoded.

Entropy coding (EC) achieves additional compression losslessly by encoding the quantized DCT coefficients more compactly, based on their statistical characteristics. The JPEG proposal specifies both Huffman coding and arithmetic coding. The baseline sequential codec uses Huffman coding, but codecs with both methods are specified for all modes of operation. Arithmetic coding, though more complex, normally achieves 5–10% better compression than Huffman coding.

Since JPEG compression was designed for photographs with smooth continuous tones, it does an excellent job with these types of images. It does a poor job, however, with images composed largely of solid color blocks and lines; it tries to shade them and smooth them out, resulting in foggy surfaces. When saving an image as a JPEG, most software allows you to select a quality level. The higher the quality, the less compression and image degradation, but the higher the file size. As always, you must find the right tradeoff between image quality and file size.

2.2. DjVu

DjVu, pronounced ''déjà vu'' in French, and ''deija voo'' in English, is an image compression technology specifically designed for scanned document pages such as books, magazines, catalogs, newspaper articles, technical publications and ancient and historical documents as well as digital photos [12].

DjVu typically achieves compression ratios about 5–10 times better than existing methods such as JPEG

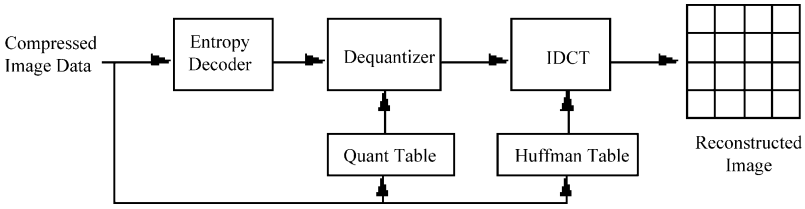


Fig. 2. JPEG decoder block diagram.

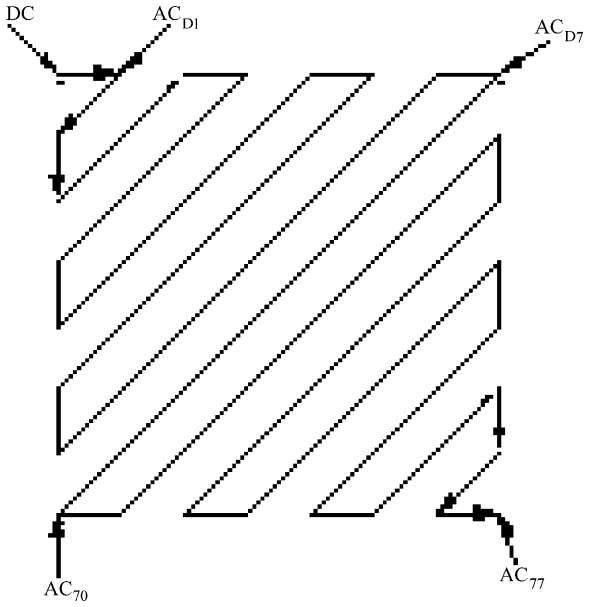


Fig. 3. Zig-zag sequence.

and GIF for color documents, and 3–8 times than TIFF for black-and-white documents. Scanned pages at 300 DPI in full color can be compressed down to 30–100 KB files from 25 MB. Black-and-white pages at 300 DPI typically occupy 5–30 KB when compressed. This puts the size of high-quality scanned pages within the realm of an average HTML page (which is typically around 50 KB).

For color document images that contain text and pictures, DjVu files are typically 5–10 times smaller than JPEG files at similar quality. For black-and-white pages, DjVu files are typically 10–20 times smaller than JPEGs and 5 times smaller than GIFs. DjVu files are also about three to eight times smaller than black-and-white PDF files produced from scanned documents (scanned documents in color are impractical in PDF).

In addition to scanned documents, DjVu can also be applied to documents produced electronically in formats such as Adobe's PostScript or PDF. In that case, the file sizes are between 15 and 20 KB per page at 300 DPI.

The DjVu plug-in is available for standard web browsers on various platforms. The DjVu plug-in allows for easy panning and zooming of document images. A unique, on-the-fly decompression technology allows images that normally require 25 MB of RAM to be decompressed to require only 2 MB of RAM.

Conventional image viewing software decompresses images in their entirety before displaying them. This is impractical for high-resolution document images since they typically go beyond the memory capacity of many PCs, causing excessive disk swapping. DjVu, on the other hand, never decompresses the entire image, but instead keeps the image in memory in a compact form, and decompresses the piece displayed on the screen in real time as the user views the image. Images as large as 2500×3300 pixels (a standard page image at 300 DPI) can be downloaded and displayed on very low-end PCs.

The DjVu format is progressive. Users get an initial version of the page very quickly, and the visual quality of the page improves progressively as more bits arrive. For example, the text of a typical magazine page would appear in just 3 s over a 56 Kbps modem connection. In another second or two, the first versions of the pictures and backgrounds will appear. Then, after a few more seconds, the final full-quality version of the page is completed.

One of the main technologies behind DjVu is the ability to separate an image into a background layer (i.e., paper texture and pictures) and foreground layer (text and line drawings).

Traditional image compression techniques are fine for simple photographs, but they drastically degrade sharp color transitions between adjacent highly contrasted areas, which is why they render type so poorly. By separating the text from the background, DjVu can keep the text at high resolution (thereby preserving the sharp edges and maximizing legibility), while at the same time compressing the background and pictures at lower resolution with a wavelet-based compression technique.

3. The underlying neural network models theory

The adaptive vector quantization theory (AVQ) is one of the most recent techniques used in the domain of image compression [2–5]. In the AVQ image compression-based approach, the input image is divided into equal size sections (subimages), where each section of n^2 pixels is considered as a vector in

the encoding space $N^{n\times n}$. The neural net clusters the subimages into classes of similar subimages. Each class has a representative vector, the *centroid*, that represents any of its member subimages. All centroid representatives for an image are to be tabled in a lookup table or *codebook*. Then, each subimage is compressed into (i.e., encoded as) the index of its corresponding class representative in the codebook. Thus, the compressed image is a set of representative indices that represent (in order) its original subimages. The compression is realized because the byte size of the original subimage (n^2) is an order of magnitude larger than its corresponding centroid bit index ($\log_2$ |codebook = number of classes|).

Since compression ratio and image quality are two very critical and conflecting factors, a convenient way to gracefully compromise between them is needed [9]. In the local codebook approach, each image will have its own codebook, even when the codebooks of same domain images overlap. Motivated by the great overhead of extra codebook per image and the codebooks overlap, we introduced the universal codebook approach [10]. In the new mechanism, all images in the same domain will train one universal codebook, increasing the compression ratio. Unfortunately, the image quality will be affected, depending on the codebook quality. A rich universal codebook, trained long and hard, might result in the inclusion of the most essential centroids, for good recovered image quality.

The Kohonen self-organizing feature map is a structured network with a two-dimensional array of nodes in which an adaptive partner weight vector is associated with every node [2]. Based on the AVQ theory, the KSOFM initializes the weight matrix randomly. The input vectors are presented sequentially to the network [1]. During the training process, the nearest weight vector (winner) to the input vector is adaptively moved closer to such an input vector. In addition, adjacent centroid that belongs to certain neighborhood space, around the winning centroid, are also trained [8]. Initially, the neighborhood radius is very large, then it decreases as the training progresses. Consequently, with every introduction of an input vector, the winning vector moves toward some theoretical class's center of mass (centroid) in its way to become the true representative of all input vectors that have won and so are similar. The input vector is associated with the class of its corresponding representative centroid. The input vectors are presented in epochs until no change of memberships is observed.

The ART1 was developed by Carpenter and Grossberg (1987, 1988) [6–8]. It serves the purpose of cluster discovery. Like networks using a single Kohonen layer with competitive learning neurons, this network learns clusters in an unsupervised mode. The novel property of the ART1 network is the controlled discovery of clusters. In addition, the ART1 network can accommodate new clusters without affecting the storage or recall capabilities for clusters already learned (elasticity/plasticity property). Essentially, the network ''follows the leader'' after it originates the first cluster with the first input pattern received. Then, it creates the second cluster if the dissimilarity between the incoming input pattern with the first input pattern exceeds a certain threshold (failed pass the vigilance test); otherwise the pattern is classified with the first cluster.

These features of elasticity and a single (one epoch) training cycle truly characterize the ART model.

4. Direct classification approach

Based on the AVQ theory, we designed our new direct classification neural net engine for image compression/decompression [1]. It follows the ''winner-takes-all'' feature of the Kohonen model, and the elasticity and single epoch training cycle of the ART1 model.

4.1. Training/compression phase

As shown in Fig. 4, the DC's training sub-phase starts by dividing the total subimage vectors domain into classes (clusters) of subimages. The class centroid represents the center of mass (average) of all of its class member subimages, and its index in the image codebook will be used to replace any member subimage in the compressed image file. For each input subimage *S*, we adaptively train the DC synaptic weights based on the KSOFM ''winner-takes-all'' concept. The process of adjustment starts with finding the best matching set of possible winning centroids (PWC) that do not exceed a certain number of

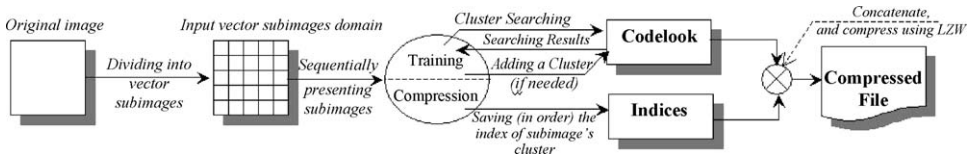


Fig. 4. DC training/compression phase.

acceptable differences with S. Then, among the PWC, we obtain the winning centroid, WC with the smallest mean error with S. Finally, WC is adjusted based on S in order to keep it representing the average of all of its represented class members, including the new member S. The failure to find PWC forces the system to add a new cluster with S as its first member and also its centroid. In case of running out of space to add entries in the codebook, the system will consider the entire codebook as the PWC of S.

The compression sub-phase assigns the corresponding WC's index (in the codebook table) as the S compressed value. The set of the total original subimages' indices will be stored in the same sequence they were introduced to the compression engine, yielding the compressed image file. In order to achieve a better compression ratio, LZW is performed on the compressed file before final storage (or transmission).

Step 1: Parameters setup.
CS: Codebook size (maximum number of entries).
IT: Intensity threshold representing the maximum difference between two corresponding vector coordinates.
TSS: Training set threshold (size) representing the maximum number of subimages allowed to adjust the center of mass of their cluster.
Step 2: Initialization.
COE $\leftarrow$ 0; number of codebook occupied entries.
Codebook: $\{C_i | i = 1, 2, \ldots, \text{COE}\}$ (initially empty).
$\{\text{SCS}_i \leftarrow \phi$ (ith set of classified processed subimages$)|i = 1, 2, \ldots, \text{CS}\}$.
Step 3: Divide the input images into subimages of equal size (n^2).
Step 4: Present an input subimage S to the system.
Step 5: For each center of mass C_i (ith centroid in the codebook), generate a distance vector D_i:

$$D_i \leftarrow |S - C_i|, \quad \text{for} \quad i = 1, 2, \ldots, \text{COE}$$

Step 6: For each center of mass C_i, compute ND, the number of D_i coordinates exceeding IT.
Step 7: Form PWC, set of possible winner centroids C_i, whose ND = 0 (most similar centroids).
Step 8: If PWC is empty go to Step 8a. Otherwise, select the best match (nearest) centroid C_j from PWC according to the mean square criterion:

$$D_j \leftarrow \min_{i=1,2,\ldots|\text{PWC}|} (D_i)$$

Go to Step 9.
Step 8a: If the codebook is full (COE = CS), assign PWC to be the whole codebook (all centroids) and go back to Step 8. Otherwise, add a new cluster, containing only the subimage S:

$$\text{COE} \leftarrow \text{COE} + 1$$
$$\text{SCS}_{\text{COE}} \leftarrow \text{SCS}_{\text{COE}} \cup \{S\}$$
$$C_{\text{COE}} \leftarrow S$$
$$j \leftarrow \text{COE}$$

Step 9: If $|\text{SCS}_j| = \text{TSS}$ go to Step 10. Otherwise, adjust the center of mass C_j:

$$C_j \leftarrow \frac{|\text{SCS}_j| C_j + S}{|\text{SCS}_j + 1|}$$

Add the subimage S to the cluster of C_j:

$$\text{SCS}_j \leftarrow \text{SCS}_j \cup \{S\}$$

Step 10: (Compression sub-phase) save the index j to be the corresponding entry of S in the compressed file:

$$\text{win}_S \leftarrow j$$

Step 11: If there are more subimages to train, go to Step 4.
Step 12: Save the winner indices (win's) and the codebook entries (C_i) into the compressed file.
Step 13: Compress further the compressed file using LZW, GZIP, WINZIP, etc.

The above algorithm is suitable for local codebooks (one codebook per image). In case of training a un-

iversal codebook (one codebook serves many images), the training phase and the compression phase are separated. The training phase will be performed excluding Steps 10, 12 and 13 from the above algorithm, and adding a final step of saving the centroids in a separate codebook file. The compression phase is also performed as mentioned above, except the centroid adjustment part (Steps 8a and 9), and excluding the centroids saving in Step 12.

4.2. Decompression phase

At the decompression phase (Fig. 5), LZW decompression is performed on the compressed file, yielding a file of indices. Then, for each index, a lookup process is performed in the codebook to obtain the corresponding centroid representative of the original subimage. The obtained centroids are placed, in sequence, at the decompressed file.

Step 1: Load the compressed file. Decompress it using LZW, GZIP, WINZIP, etc. Retrieve the indices and the codebook.
Step 2: Select, in order, an index i from the indices tables.
Step 3: Using i as an address, access the corresponding codebook entry to obtain a centroid and store it—in the same order of index i—into the decompressed file.
Step 4: If there are more indices, go to Step 2.

4.3. DC Comparison with KSOFM and ART

The advantage of the DC over Kohonen is that the input domain is presented only once to the former (i.e., no epochs) for faster training. Therefore, the asymptotic training time complexity of the DC is $O(n)$, where n is the size of the input domain of subimages; a huge reduction of the original KSOFM time complexity of n^2. The traditional ART1 [6,8] generates small real numbers in the weight matrix, which might cause the deviation of the classification, in case of low system precision. Instead, our DC model develops classes' representatives (centers of mass) called centroids, which are of the same type as the input subimages (vectors of integer coordinates).

ART and SOFM have a major difference in the number of neurons to be trained. At any stage of SOFM, the number of neurons is fixed, whereas, in ART, a neuron is added when the input does not match (within certain threshold) any of the existing centroids [14]. The use of fixed number of neurons results in faster convergence and lesser bookkeeping, while techniques based on varying the number of neurons have more accurate results for smaller input set. Some hybrid models divide the training phase into multiple cascaded phases, where in each phase the network is trained with either SOFM or ART. For instance, in order to solve the traveling salesman problem (TSP), Vishwanathan and Wunsch proposed a hybrid model to cluster cities in three phases [14]. In the first phase, ART is used to form clusters of cities. In the second phase, a modified version of SOFM is utilized to solve the TSP inside each cluster. In the third phase, all the clusters are connected to form the optimal path (solving the TSP problem) via the use of a different version of SOFM. This ART/SOFM hybrid is performed sequentially, where each phase feeds its output to the next phase, resulting in chained training. In contrast, our DC model is neither ART nor SOFM. Instead, it carries out a one training epoch only (out of the many epochs of SOFM), in order to speed up the convergence process. Experimental results showed that the choice of the right thresholds (i.e., CS, IT, and TSS) and the initial value of the centroid (as a complete subimage) was enough to obtain very satisfactory image quality (PSNR), with only one epoch. Moreover, DC reduces the SOFM to the simple competitive learning rule of Kohonen by

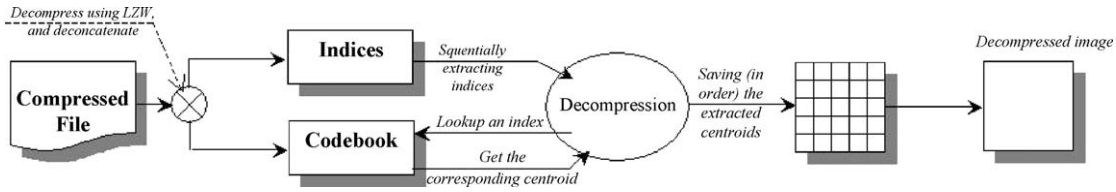


Fig. 5. DC decompression phase.

setting the neighborhood radius to zero. When the input is presented to the DC training engine, the centroid to be adjusted is selected based on the ART model (Steps 7 and 8 in the DC training phase), but only the centroid modification formula is based on SOFM (Step 9 in the DC training phase). As we will discuss next, in Section 6.4, the formation of the centroid in the DC model will involve only a TSS members of its class, instead of the entire class members as in KSOFM.

4.4. DC comparison with wavelet

Our work is in the same area as the well-known wavelet image compression technique, with a major difference in the representation of the compressed image, reflected in the formation of the lookup tables [1].

In principle, the discrete cosine transform phase of the wavelet introduces no loss to the source subimages; it merely transforms them to a domain in which they can be more efficiently encoded (coefficients). After the DCT phase, each of the encoded coefficients is uniformly quantized in conjunction with a carefully designed quantization table [13]. At the decoder, the quantized values are multiplied by the corresponding QT elements to recover the original unquantized values (original subimages). In order to achieve the target compression, wavelet uses the entropy coding to encode the quantized DCT coefficients more compactly, based on their statistical characteristics, yielding lossy compression.

On the other hand, the manufacturing of the DC's lookup tables is carried out via the training of a hybrid neural model of the SOFM and ART nets, with some modifications. Our model is a vector quantizer (VQ), which encodes subimage vectors via the mapping of many similar k-dimensional input vectors (with respect to a given distortion measure) into one representative codeword (centroid) vector [3]. The similarity measure of vectors X and Y is based on the distance $|X - Y|$ (distortion) between X and Y. A collection of centroids is stored in a lookup table (codebook), which is utilized later, at the decompression phase in the lookup process. The manufacturing and the nature of the codebook are the key distinctions of our work from other peer mechanisms.

5. Image characteristics

Image quality and image compression ratio are two contradicting factors. Since the compressed image is composed of a codebook and an indices file. When the number of classes (NC) increases, the codebook linearly increases, and the indices' size logarithmically increases, yielding a lower compression ratio. Decreasing the number of subimage members in a class (fine-tuning) improves the accuracy of the centroid representation of its class members, since the centroid is just their average. In fact, the image quality is proportional to the centroids' representation accuracy. Since the number of subimages is fixed, decreasing the number of subimages per class will increase NC. Hence, the image quality is a tradeoff with the compression ratio via the NC parameter. Next, we present universal metrics to measure the quality and compression ratio.

5.1. Quality measurement

The quality of the image is one of the most important criteria. Actually, many schemes have been proposed for the measurement of such criterion, yet the optimal or the most significant definition is still debatable. One of the measurements is the difference of the intensity between the original image and the decompressed image. The most common measures specified for this purpose are:

(i) Mean squared error: $\mathrm{MSE} = \sum_{i=1}^{N} \sum_{k=1}^{\mathrm{color}} (x_{ik} - x'_{ik})^2 / (N \times \mathrm{color})$, where N is the number of pixels; color: 1 for gray scale and 3 for colored images; x_i and x'_i, are the original byte and its corresponding recall.

(ii) Peak single noise to ratio: $\mathrm{PSNR} = 20 \log_{10} (255/\sqrt{\mathrm{MSE}})$. Usually a value higher than 30 reflects a good quality, and it is hard to detect image degradation visually.

5.2. Compression ratio

$$\text{Compression ratio} = \frac{\text{original image size}}{\text{compressed file size}}$$

This ratio is always related to ''one'' unit; e.g., if the original image size is 786,447 bytes, and the com-

pressed file is 12,563, then the compression ratio is 62.60:1.

6. Training parameters

In this section, we present the three training parameters that characterize our training model, using several colored documents (Table 2), photos (Fig. 12), and satellite images (Fig. 13) for accurate result analysis.

6.1. Codebook size

Assuming that the size of the codebook is CS, and the size of the subimage is n, the initial compression ratio achieved is $n/(\log_2(\mathrm{CS}))$. In our DC model, the codebook size is the major factor for controlling the quality of the image and the compression ratio. Fig. 6 clearly shows that the larger the codebook, the smaller the class size. A small class size means a better representation of the input subimages by their corresponding class centroid, hence better quality. However, a larger codebook requires a longer bit representation of the indices centroids, resulting in a lower compression ratio. Also, experimental results show that the compression time depends directly on the codebook size. Such dependency is due to the time spent in searching for the closest centroid to an input subimage.

6.2. Training set size

In the DC training phase, the training set size threshold (TSS) represents the maximum number of inputs allowed to adjust a specific centroid; only the first TSS members (subimages) can modify the center

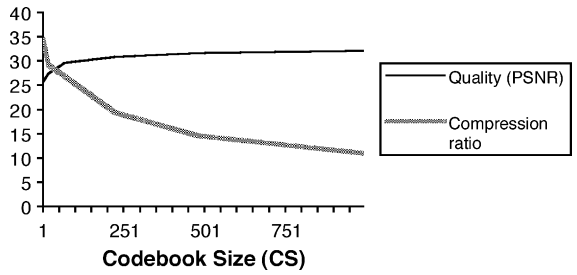


Fig. 6. Compression performance varying the codebook size.

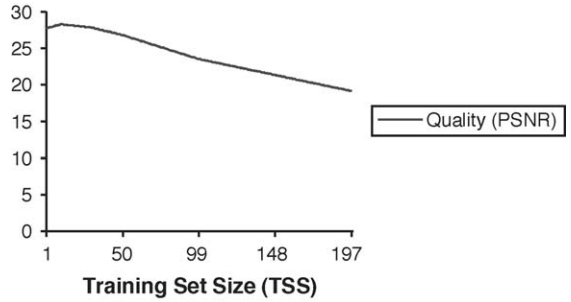


Fig. 7. Compression performance varying the training set size.

of mass. Fig. 7 clearly shows that the size of the training set (controlled by TSS) dramatically affects the quality of the decompressed image. The application of the TSS parameter does not affect the class membership; i.e., the size of the class. Hence, varying TSS will not affect the number of formed classes (i.e., the number of centroids), leading to same size codebook. Thus, the compression time and ratio remained nearly the same, since the codebook is of fixed size.

Fig. 7 shows that the quality drops linearly as TSS increases. Based on such experimental observation, we invoked only the early arriving subimages (TSS) in the process of centroid adjustment. When the codebook is filled, any consequent (late) arriving subimages, CLAS, to the class should not affect the centroid value. Our argument of discarding the involvement of CLAS in the centroid adjustment is justified based on their similarity to the already trained centroid. There are two scenarios for the CLAS. First, the CLAS is similar to the class centroid; in such case, it will have a minor effect in updating the centroid's value. Second, it is far away from the centroid (within the IT threshold), hence, it is not desirable to be updating the centroid, because of the negative effect on the good representation of the centroid to all other subimages that are similar to the centroid. Honoring the core of the class member representation is more important than the few sparse members at the class edge.

6.3. Intensity threshold

In order to achieve better quality, the input subimage vector is compared to the previously formed centroids to select the closest (most similar) as its

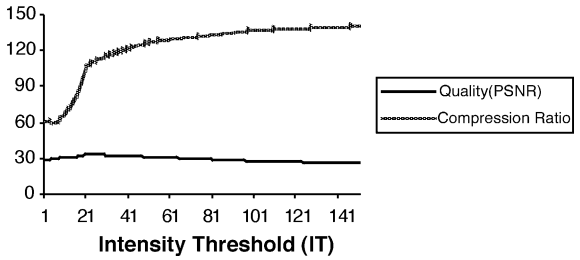


Fig. 8. Compression performance varying the intensity threshold (IT).

Table 1
Training parameter values

Image type	Number of used images	Optimal value of TSS	Optimal value of IT
Gray-scale images	50	32	2
Non-document colored images	30	5	32
Document colored images	100	10	64

representative. The IT threshold controls the comparison between corresponding coordinate bytes of the subimage and each compared centroid. Fig. 8 shows that the quality increases around the values between 15 and 25, but not at high or low values. Small values of IT will result in a more accurate membership at the early stages; hence, filling the codebook quickly, creating more classes, i.e., decreasing the compression ratio. After exhausting the codebook entries, incoming subimages will be added in an ad hoc approach, which might result in degrading the quality. On the other hand, large values of IT will permit "many" subimages to belong to the same class, decreasing the number of classes, which increases the compression ratio. But, lumping large numbers of subimages in a class with large variance from the class centroid decreases the image quality.

We do not have a mathematical formula to relate the system critical parameters (CS, TSS, and IT) to the optimal balance between quality and compression ratio. Yet, we are able to obtain experimentally an acceptable approximation of their values, as shown in Table 1. In such experiments, we fixed CS (for a desired compression ratio); then, we tried all possible IT and TSS values in order to achieve the maximum quality.

The training process involves the sequential scanning of the codebook, for every input subimage, i.e., it has a time complexity of $O(L \times CS)$, where L is the number of input subimages. Since the CS is fixed, the time complexity reduces to $O(L)$, which is much faster than $O(L^2)$ of KSOFM.

In practice, experiments showed compression time ranging from 2 s to 2 min of same L. In order to improve the codebook search time, we tried to sort the formed centroid, based on their values. As a result, the compression speed dropped (2–6 s), yet there was a serious drop in the decompressed image quality. Since the quality and the compression ratio are more critical in our non-real-time environment, we did not consider the compression time to be critical.

6.4. DC degradation

A very complicated image (that contains large variety of non-similar subimages) requires a large codebook, in order to maintain good quality. For instance, our DC compression suffers when the image contains gradually shaded regions (Fig. 9), which requires a precision of class fine-tuning (smaller size classes) resulting in larger codebooks.

7. DC application in the colored image domain

We experimented with a wide variety of different images. We drew intensity graphs and pixel value distributions in order to find a balance of the training

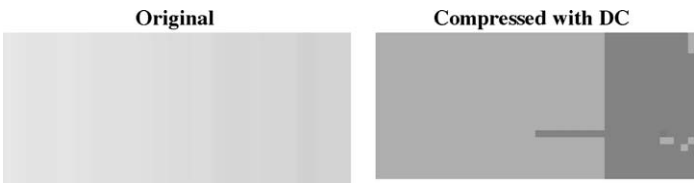


Fig. 9. DC degradation in gradually shaded regions.

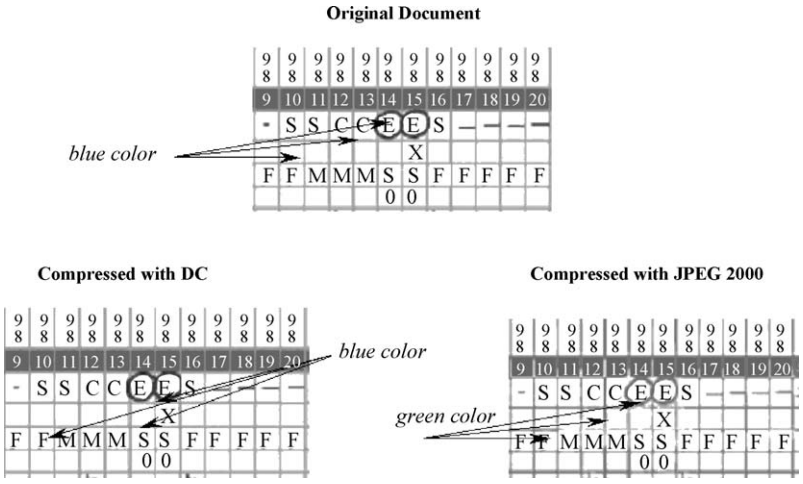


Fig. 10. DC's perfect compression vs. color degradation in DjVu's compression.

parameters that achieve good performance. A simple conclusion seems very difficult to achieve after a considerable number of experiments using our DC model. The lack of common characteristics among images, even of the same domain, led us to try different parameter values for each image.

We chose DjVu in comparison with our DC since the former is the best existing compression technique vis-à-vis documents. Although DjVu's developers claim that it treats the background of the image (especially documents) separately, experiments showed that DC is better at background recognition [12]. For instance, DjVu suffers from the run of color in some documents as shown in Fig. 10 (blue changing to green), whereas DC preserved the same quality of the document, with higher compression ratio. The DC recognizes the background as the centroids' representatives of the highest prominent subimages. Placing such centroids near the start of the codebook (index 0, 1, 2, . . .) will guarantee many zeros in their binary representation. This will lead the indices files to have many runs of zeros since those centroids' indices are the most repeated inside the compressed file. The next table shows the cases where good results were obtained. Thus, using LZW will result in significant improvement in the compression ratio.

In Table 2, DC achieves a better compression ratio than DjVu (most of these documents are listed in [16]). DjVu failed to surpass the DC model in the domain of non-purely-textual documents (mostly graphs and figures). In some examples, as in Cdoc12, the compression ratio was nearly three times that of the DjVu, with a lossless restoration of the original file. Such excellent results are achieved since the number

Table 2
Document compression results using DC and DjVu (CR: compression ratio)

Image	DjVu		DC	
	CR	Quality	CR	Quality (PSNR)
Cdoc4	58.70	Good	73.76	32.58
Cdoc6	82.63	Good	89.45	32.83
Cdoc7	63.19	Good	65.70	34.06
Cdoc10	94.06	Good	164.74	34.65
Cdoc12	58.15	Good	156.79	Lossless
Cdocl5	50.71	Foggy	54.28	34.78
Cdoc19	83.30	Good	91.85	36.67
Cdoc21	88.38	Good	92.19	34.22
Cdoc22	48.96	Good	70.46	34.02
Cdoc23	120	Good	122.01	36.76
Cdoc24	84.40	Good	85.81	31.53
Cdoc26	41.06	Foggy	80.15	31.99
Cdoc27	189	Good	192.05	31.75
Cdoc32	48.46	Foggy	53.63	34.27
Cdoc39	77.31	Good	136.89	32.83
Cdoc41	28.99	Foggy	74.50	32.20
Cdoc42	95.03	Foggy	207.23	31.44
Cdoc43	36.06	Good	61.29	31.30
Cdoc44	87.30	Good	118.37	31.53
Cdoc45	51.15	Foggy	53.57	30.33
Cdoc47	58.27	Foggy	70.25	31.58
Cdoc48	28.30	Foggy	61.72	31.36

Table 3
Non-document image compression using DC model

Image	Compression ratio	Quality (PSNR)
Arcticfox.ppm	16.44	34.41
Cat.ppm	16.31	28.46
Lena.ppm	19.33	30.84
Philip.ppm	23.57	30.50
Arch.ppm	16.89	29.76
Boat.ppm	17.12	29.22
Sat2.ppm	16.86	28.55
Sat4.ppm	19.30	24.40
River.ppm	21.49	27.73
Bear.ppm	17.15	27.76
Horse.ppm	14.77	31.68

All the experiments were done based on the following parameter values: CS = 256 centroids, IT = 5, and TSS = 32 subimage inputs.

of different subimage patterns inside the original image is small. The ability to tune the quality and the compression ratio is maintained in the DC model. Such flexibility is achievable due to the private generation of the codebook for each image.

When the document is textual, i.e., mostly lines of characters with well-known fonts, DjVu could do better in both quality and compression ratio. The reason is that DjVu uses some character recognition techniques [12], which represent a character in few bytes. DC model needs in average 56 bytes to represent a character matrix of 8×7 pixels. Thus, pure textual documents are not good candidates for our model.

7.1. Non-document color images

For non-document color images, as shown in Table 3, DC model yields some acceptable PSNR, but the compression ratio is mostly lower than that of the document domain. However, in case of images of only solid color background, our DC yields a much higher compression ratio than peer techniques (Fig. 11).

8. Geosynchronous satellite image domain: DC universal codebook application

Due to the great overhead of extra codebook per image and the codebooks' overlapping, we utilize the universal codebook approach in the compressing and decompressing process. Therefore, all images of the same region will be used to train one universal codebook, yielding an increase in the compression ratio; in contrast with the local codebook, the compression ratio is related to the size of the indices file only, excluding the codebook. The image quality will depend on the precision of the codebook training. A rich and long universal codebook will include the most important centroids, for good recovered image quality. Theoretically, the generated universal codebook will not be counted against the compression ratio; it will be stored at the sender and the receiver sites instead.

We carried out many experiments using different satellite images in order to train several universal codebooks. The result of growing huge universal codebooks is a larger image index size, yielding a lower compression ratio. To solve the problem, we divided the image into regions, with each having its own small codebook. The total universal codebook is the integration of all such small regional codebooks. Every image is divided into regions (1000 regions),

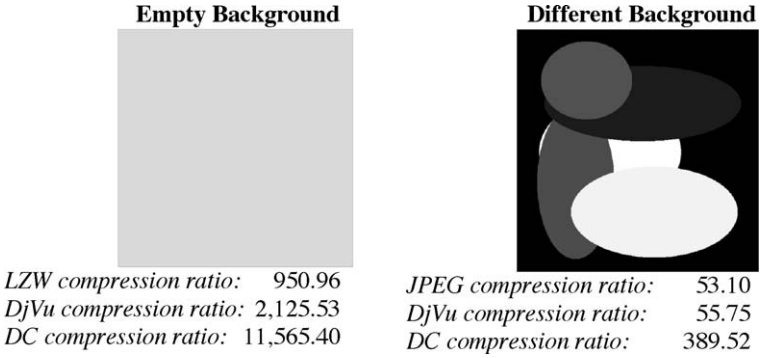
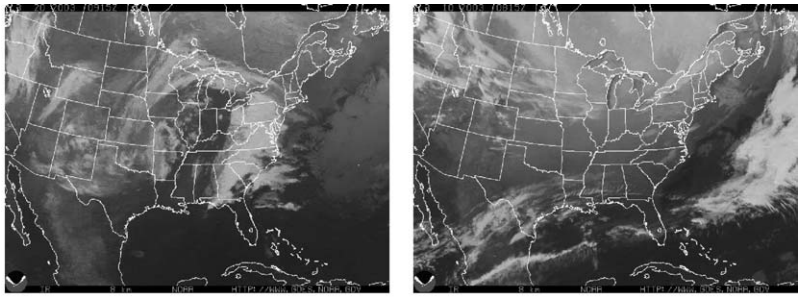


Fig. 11. DC outperforms JPEG and DjVu in solid backgrounds.

Fig. 12. Compression of "Lena" using DC.



Decompressed image using DC (image from the training set)

DC compression ratio 57:1
JPEG2000 compression ratio 35:1

Decompressed image using DC (image out of the training set)

DC compression ratio 42:1
JPEG2000 compression ratio 35:1

Fig. 13. Satellite image compression using DC.

where the *i*th region trains the *i*th regional codebook. Such mechanism allows every regional codebook to learn about similar smaller regions from different images; hence storing small number of centroids, yet enough for good image quality. The small size of the regional codebook helped also in decreasing the local search time (15 min to train 100 images, and 6–15 s to compress an input image, @ 333 MHz PC).

We utilized intensity graphs and pixel value distributions in order to find a balance of the training parameters CS, TSS and IT that achieve good performance, balancing high image quality and compression ratio. Experimental results showed such balance at TSS = 200, IT = 15, and CS = 256.

There are two sets of experimental results of our DC model: a set of results obtained from the

Table 4
Satellite image compression results using DC

Trained images			Non-trained images		
Image	Compression ratio	Quality (PSNR dB)	Image	Compression ratio	Quality (PSNR dB)
Sat1.ppm	58.16	38.70	Sat101.ppm	42.85	30.66
Sat2.ppm	57.85	38.49	Sat102.ppm	42.97	30.89
Sat3.ppm	56.67	37.57	Sat103.ppm	42.74	31.01

compression/decompression of the same set of images used to train the universal codebook, and images outside the training set. As shown in Table 4, when processing the training set images, we obtained a 70% increase in the compression ratio over the peer wavelet/JPEG2000 models, for the same quality. But, for images outside the training set, we obtained acceptable quality (above 30 dB S/N ratio, visually undetectable degradation), with a slight increase (about 30%) in the compression ratio over existing peer mechanisms (Fig. 13).

9. Conclusion

The DC hybrid approach using the two prominent neural models, Kohonen and ART1, proves to be a promising image compression model. The major sources of DC power are the utilization of some features of the ART1 (fast training) and the KSOFM's reliable/accurate centroid formation, in addition to the DC's own special features (e.g., introduction of tuning parameters). Experimental results show that our DC model is very competitive with other well-known peer techniques (e.g., JPEG2000 and DjVu wavelet technology) in the document and satellite imaging domains. Moreover, the DC neural net model can train a ''universal'' instead of a ''local'' codebook (for a single homogenous domain of images) for better compression performance, even for images outside the training domain. In secure satellite applications (e.g., military domain), in addition to saving the bandwidth via compression, the DC's inherent privacy (of the compressed file) saves the encryption/decryption overhead of the transmitted images. Our future work will involve the deployment of the DC model into other image domains, e.g., pictorial (photos), landscape (rivers, mountains, waterfalls, etc.), and video streaming.

References

[1] H.S. Soliman, M. Omari, Image Compression using Neural Nets, Hybrid ART/Kohonen Neural Model for Document Image Compression, ANNIE 2002, University of Missouri-Rolla, MO, November 2002.
[2] T. Kohonen, Self-Organizing Maps, Optimization Approaches, Artificial Neural Networks, Elsevier/North-Holland, 1991 pp. 981–990.
[3] T. Kohonen, A program package for the correct application of learning vector quantization algorithms, in: IEEE International Joint Conference on Neural Networks, 1992.
[4] D. Hankerson, G.A. Harris, P.D. Johnson Jr., Introduction to Information Theory and Data Compression, CRC Press, 1998.
[5] E. Erwin, K. Obermayer, Self-organizing maps: ordering, convergence properties and energy functions, Biol. Cybern. 67 (1992) 35–45.
[6] N.M. Nasrabadi, A.K. Katsaggelos, Applications of Artificial Neural Network in Image Processing III, San Jose, California, April 1998.
[7] N.M. Nasrabadi, A.K. Katsaggelos, Applications of Artificial Neural Networks in Image Processing V, San Jose, California, 2001.
[8] J.M. Zurada, Introduction to Artificial Neural System, PWS Publishing Company, 1995.
[9] S. Ramakrishnan, K. Rose, A. Gersho, Constrained-Storage Vector Quantization with a Universal Codebook, IEEE, 1998.
[10] H.S. Soliman, M. Omari, Universal Codebook versus Local Codebook: Applications of Image Compression Using AVQ Theory, ANNIE 2002, University of Missouri-Rolla, MO, November 2002.
[11] R. Hecht-Nielson, Counterpropagation networks, Appl. Opt. 26 (1987) 4979–4984.
[12] Answers to frequently asked questions about DjVu, http://www.djvuzone.org/support/faq.html.
[13] Image compression from DCT to wavelets: a review, http://www.abmedia.dk/video/coding.htm.
[14] N. Vishwanathan, D.C. Wunsch, ART/SOFM: a hybrid approach to the TSP, in: INNS–IEEE International Joint Conference on Neural Networks (IJCNN'01), vol. 4, 15–19 July 2001.
[15] The Art of Scientific Computing, Cambridge University Press, 1986–1992.
[16] Examples of the application of direct classification approach in image compression, www.cs.nmt.edu/~hss/dc.htm.