Journal of Applied Sciences 11 (17): 3152-3160, 2011
ISSN 1812-5654 / DOI: 10.3923/jas.2011.3152.3160

Codebook Enhancement in Vector Quantization Image Compression using Backpropagation Neural Network

Omaima N.A. AL-Allaf
Department of Computer Information Systems, Faculty of Sciences and Information Technology,
AL-Zaytoonah Private University of Jordan, P.O. Box 130, Amman (11733), Jordan

Abstract: Vector Quantization (VQ) is a powerful technique for image compression. One of the most VQ problems is the high computational complexity of searching for the closest codevector in the codebook during the compression phase. Artificial Neural Networks (ANNs) were used in image compression where high computational performance is required. A three layered Back Propagation Neural Network (BPNN) was proposed in this research for building an enhanced codebook for VQ compression of images. The Backpropagation neural network algorithm (BP) was used for training the designed VQ BPNN. Finally, a trained VQ BPNN was obtained to produce the codevectores of any image by the hidden layer neurons. We can later apply both trained and un-trained images to this VQ BPNN to compress them. Experiments were conducted with different VQ BPNN architecture and BP parameters to speed up this algorithm and enhance VQ codebook. It is observed that proposed algorithm is faster than other algorithms, although it needs a time for learning process. The performance of VQ BPNN image compression can be increased by modifying the VQ BPNN architecture especially, the number of hidden layer neurons and modifying BP learning parameters.

Key words: Image compression, vector quantization, artificial neural networks, backpropagation neural network, backpropagation algorithm

INTRODUCTION

Artificial Neural Networks (ANNs) are composed of many interconnected non-linear elements that operate in parallel and emulate the parallel distributed processing of the human nervous system (Gholizadeh and Darand, 2009). The Back Propagation Neural Network (BPNN) is the most widely used ANN architecture and it consists of three or more fully interconnected layers of neurons. The BPNN can be trained using back propagation algorithm (BP). The BP can be applied to multilayer network that uses differentiable activation function and supervised training (Al-Daoud, 2009). ANN has found to be a reliable classification tool especially in the image and signal processing (Ibrahim *et al*., 2005). The BPNN is the most widely used network in image processing.

Image compression using Vector Quantization (VQ) has received a lot of attention because of its simplicity. VQ requires the input image to be processed as vectors or blocks of image pixels (Jilani and Sattar, 2010). Image VQ includes four steps: vector formation, training set selection, codebook generation and quantization. The first step is the decomposition of input image into a set of vectors. A subset of vectors in the set is later chosen as a training sequence. The codebook of representative vectors (codewords) is generated by using an iterative clustering algorithm. Finally, in quantizing an input vector, the closest codeword in the codebook is determined and the corresponding label of this codeword is transmitted. In this process, data compression is achieved because address transmission requires fewer bits than transmitting vector itself (Boopathy and Arockiasamy, 2010).

Two algorithms that are most widely used in VQ codebook design: LBG clustering algorithm (Linde *et al*., 1980) and Equitz Nearest Neighbour algorithm (ENN). The LBG disadvantages: poorly selected initial codebook may lead to undesirable final codebook and a complete design requires a very large number of computations. The computation associated with LBG algorithm is proportional to the training data size and codebook size. For example, 128×128 image partitioned into 4×4 blocks has 1024 training block vectors. The LBG algorithm compares each training block against all codewords in the codebook and then decides the closest match. This procedure is repeated for every training vector.

While ENN algorithm eliminates the need of an initial codebook and reduces the computations dramatically. At the beginning of ENN algorithm, all training vectors are viewed as initial clusters (codevectors). Then, the two nearest vectors are found and merged by taking their average. These two vectors are replaced by new vector

reducing number of clusters by one. This process is repeated until reaching the desired number of clusters. If all possible pairs are to be compared at each merging step, the total computations are numerous. A new procedure is introduced for reducing the number of comparisons. The performance of ENN algorithm is better than that of the LBG algorithm with a randomly selected initial codebook. Better codebook can be constructed by first using Equitz algorithm to obtain the initial codebook and then using the LBG algorithm to refine the codebook (Bayir, 2008). But this process requires a very large number of computations and this require a very long time.

One of the major VQ drawbacks is its complexity in encoding, search time and the memory space required for storing encoder/decoder codebooks. Therefore, many VQ techniques have been investigated to reduce the search and storage complexities of the codebook (Bayazit and Pearlman, 1999). Somasundaram and Domnic (2006) proposed image compression model by compressing VQ indices and generating residual codebook. The VQ indices are compressed by exploiting correlation among image block to reduce the bit per index. A residual codebook is similar to VQ codebook which is generated to represent the distortion produced in VQ. Using this residual codebook, the distortion in the reconstructed image is removed. Their results on image Lena give a reconstructed image with a PSNR value of 31.6 db at 0.396 bits per pixel.

Zhe-Ming and Yue-Nan (2010) proposed an algorithm called mean value predictive VQ (MVPVQ) to reduce the computational complexity. Input vectors are classified into smooth vectors and non-smooth vectors before encoding. In their method, different kinds of input vectors are encoded with different encoding schemes to achieve satisfying image quality at low bit rates. Their results give a reconstructed image with a PSNR value of 29.774 db at 0.562 bits per pixel.

Many researches in the literature addressed the use of ANNs in image compression to reach the technique that lead to high compression ratio with better image quality. Many ANN models have been used for VQ of images. These methods are concerned with the search for good codebooks and speed up the computation of VQ encoding/decoding process (Narayanan and Baboo, 2010). One of these models is called the Self-Organizing Feature Map (SOFM) and it was proposed by Kohonen (1990) as a learning algorithm for ANNs. The SOFM is an efficient way to achieve VQ design. Kohonen networks realize a mapping between an input and an output space that preserve topology. Amerijckx *et al.* (2003) used SOFM to build compression scheme to obtain better compression rate than with using JPEG classical method without reducing the image quality. As result, Lena image compressed by their proposed method, compression rate 25.22, PSNR 24.7 dB.

Rawat and Meher (2009) proposed a hybrid model combining SOFM based VQ and Set Partitioning In Hierarchical Trees (SPIHT) coding for effective image compression. First, the input image is decomposed using biorthogonal wavelet transform. The decomposed image is compressed using SPIHT encoding to result with a bit stream. For more compression, the resulted bit stream is then fed to SOFM. The SOFM generates a codebook corresponding to inputted bit stream and compresses the bit stream based on the generated codebook. The reconstruction of the original image involves linear combination of processes as in image encoding. Their results give a reconstructed Lena image with a PSNR value of 32.14 db with 0.34 bpp.

According to literature studies, we need a simple technique to enhance the codebook of VQ image compression by reducing the computation rates and memory requirements. In this research, a technique using BPNN for codebook design is applied. This technique is based on using BP to train the three-layered ANN for designing the codebook of any image to be compressed using VQ of images. The BP algorithm offers important advantages over both LBG and Equitz NN algorithms. This research focuses on enhancing the VQ codebook by changing the network architecture and BP learning parameters.

MATERIALS AND METHODS

Vector quantization of images: The VQ requires the input image to be processed as vectors by dividing it into non overlapping blocks of subimages. The encoder takes in a vector and finds the best or closest match, based on distortion criterion from its stored codebook. The address of the best match is transmitted to the decoder. The decoder accesses an entry from an identical codebook and obtaining the reconstructed vector (Nasrabadi *et al.*, 1994). Let X represents an input vector from the original image and let Y represents a codeword. The codebook contains all the available codewords and the quantizer replaces the input vector by the index of codeword that is more similar to X. This is done by using distortion measure. A full search is required to assign to X the best codeword of the codebook. This means that every codeword has to be examined to ensure that the best will be selected. This process requires a high computational load when the number of codewords is large. Several algorithms have been designed to reduce the computational load of encoding process. Torres and

Huguet (1994) suggested an algorithm which requires additional storage spaces for information used during the encoding process in addition to the storage space required for codebook. Nasrabadi *et al.* (1994) suggested an algorithm for building a super codebook to compress more than one image using this super codebbook. The super codebook was designed using LBG algorithm (Linde *et al.*, 1980) from training set of several images. But this super codebook is very large and requires more storage space to store it and very long time for encoding process.

Codebook design depends on the source image to be encoded. The source image (N×N) is partitioned into contiguous, non-overlapping, square blocks according to a predetermined block dimension (P). Each one of these blocks is a matrix of dimensions P×P and it is rasterized to one-dimensional vector. Then initial number of codewords in codebook is M, where M = (N×N) / (P×P).

The ENN algorithm is applied on these one-dimensional vectors and the codebook is generated which is a group of vectors (codewords) usually less than the number of source image's vectors but approximately representing all of them. The implementation of ENN algorithm can be summarized by the following steps:

Step 1: At the beginning of ENN algorithm, all the image vectors are viewed as initial codewords

Step 2: Find each of the two nearest codewords using Eq. 1:

$$Od(X, Y_i) = \sum_{j=0}^{k-1} | X_j - Y_{i,j} | \quad (1)$$

and merge them by taking their average, where k is the codeword length.

Step 3: Replace the two codewords by one new codeword and reduce the number of codewords by one

Step 4: Steps [2–3] are repeated until the desired number of codewords is reached

The ENN requires long time and large number of iterations to design the codebook. Therefore, to decrease the number of iterations and time required to generate the codebook, an image block distortion threshold value (dth) is calculated from the image. The previous algorithm is modified as follows:

Step 1: Determine the desired number of codewords and the maximum number of gray levels used in the image (max-gray)

Step 2: Find the distortion threshold (dth) by the following equation: dth = k × (max-gray/64), where k is the length of codeword

Step 3: Compute the distortion error between a taken codeword and the next codeword. If the distortion error is less or equal to dth, then merge these two codewords and reduce the number of codewords by one. Otherwise consider the next codeword

Step 4: Repeat step (3) until the number of codewords is equal to the desired number of codewords

Step 5: If all the codewords are compared and merged, and the resultant number of codewords remains greater than desired number of codewords, then change value of dth as follows: dth = dth + k × (max-gray / 256) and then go to step 3

VQ encoding and decoding: The VQ encoding program can be summarized as follows:

Step 1: Divide the source image to be encoded into non-overlapping blocks according to the predefined block dimension (P). Set the PD: performance distortion to zero

Step 2: Take one block (X) from the image. Find the nearest codeword (Y_i) in the codebook file by finding the minimum value of the distortion measure d (X, Y_i) using Eq. 1. Use index of this codeword and put it in compressed file

Step 3: Modify PD such that PD← PD + d(X, Y_i)

Step 4: Repeat steps (2 and 3) for the whole image blocks

Step 5: Finally, compute the average distortion (AVDis) to evaluate the compression performance, where AVD is equal PD/no-of-image-blocks

At the beginning of the VQ compressed file, there is a header block that contains information about the source image and the compression process parameters. The rest of the compressed file contains the indexes of the codevectors that represent the image blocks.

The rate of the encoder (R) in bits per pixel (bpp) can be defined as:

$$R = [(1/k) \times \log_2 N] \text{ bpp} \quad (2)$$

where, N is the codebook length and k is the vector length. For example, let the image to be encoded be of 256×256 size and let the block dimension be 8×8, then the rate of the encoder is 1/64 $\log_2$ 1024 = 0.15 (bpp). Data compression is achieved in this process because the transmission of the address requires fewer bits than transmitting the vector itself (Ping and Anastasios, 1994).

Whereas, in the decoding process, the receiver, that has a copy of the codebook, constructs the image using codewords which corresponding the received indexes in place of the original vectors. The VQ decoding program can be summarized bas follows:

Step 1: Take one index from the compressed VQ file
Step 2: Search in the codebook file for that index and get its corresponding codeword
Step 3: Derasterize the codeword from one-dimensional vector to two-dimensional one. Put that block in the appropriate location in the decompressed file
Step 4: Repeat steps (1-3) for the whole indexes in the VQ compressed file

It can be noticed that the computation required for encoding process is very large, because during the encoding process, many computations are required to find the closest codeword for each source image block, inversely, in the decoding process, it only requires searching in the codebook.

BPNN architecture for VQ of images: BPNN architecture was suggested in this research for VQ of images to obtain best results in Compression Ratio (CR) and enhancing the codebook design. The architecture of BPNN used for VQ is equivalent to the architecture of BPNN used for image compression which suggested in previous research (Omaima, 2010). Figure 1 shows the BPNN VQ image compression and decompression system. Also in this research, VQ using BPNN requires the same processes which were used in the previous research (Omaima, 2010) such as: image normalization and segmentation; initialization BPNN learning parameters and preparation of BPNN training/testing set. For this reason, the same BPNN architecture is used for both image compression and also for improving VQ of images. A BPNN with three layers (input, hidden and output layer) is used, the number of input layer neurons is equal to the number of output layer neurons and the number of hidden layer neurons is much less than the number of input layer neurons. The input layer neurons represent the original image block pixels. Whereas, the output layer neurons represent the pixels of the reconstructed image block. The hidden layer neurons are assumed to be arranged as a one-dimensional array of neurons which represent the elements of codeword. These elements are usually less than the elements of the codeword used in the typical VQ.

The learning process of VQ of images covers all layers of BPNN (input, hidden and output layer). The learning process is accomplished by using the BP algorithm and using a set of images as training patterns. This process produces an optimal VQ codebook that is used later in VQ image compression and decompression.

After learning process is finished, building the final codebook is accomplished by using the trained BPNN on a set of images. The VQ image compression process requires the input and hidden layers. But the VQ decompression process requires the hidden and output layers.

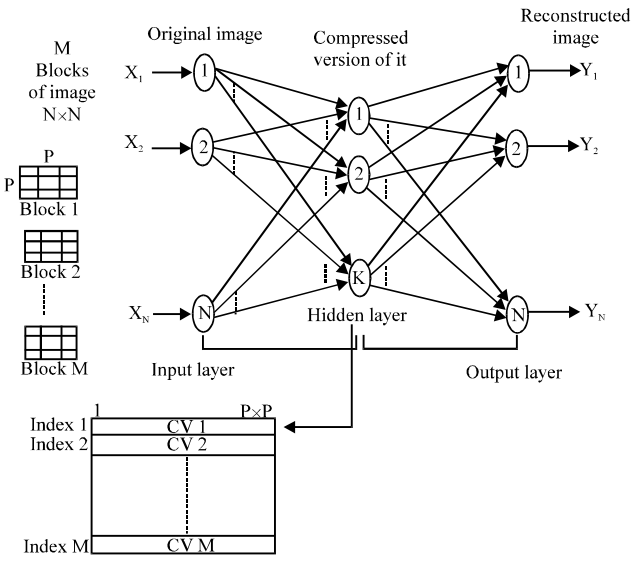

Fig. 1: BPNN VQ image compression system

The codebook as mentioned before is designed using BP. At the beginning of codebook design, the size of codeword and the size of codebook must be specified. Then the source image is partitioned into non-overlapping blocks of pixels such that the block size equals the number of the input layer neurons and the number of hidden layer neurons equals the codeword size (length). Then, the BP is used to build codebook as follows: the codebook is partitioned into rows and columns such that row = n, column = m, where n represents the number of patterns of all images and m represents the number of hidden layer units. After that, the BP learning program is applied to all images. The presentation of images blocks (training patterns) to the BPNN is repeated until convergence occurs (i.e., the total square error over all training patterns becomes less than the threshold error value). At that time, the learning process is stopped and then the BP simulation program is executed only in forward direction to obtain the outputs of hidden layer neurons for each training pattern. These outputs are saved as a codeword in the codebook file (codebook.nn) which will be used later in VQ encoding and decoding processes.

Codebook can be optimized by decreasing the number of codewords in it. The size of codebook after the codebook initialization depends on the number of training patterns of all images multiplied by the number of hidden layer units. But the size of the initial codebook can be decreased using ENN algorithm. This algorithm can be further modified to decrease the computation time.

BPNN VQ encoding and decoding: In this research, a simulation program was written in C# programming language to implement BPNN for VQ of images. The objective here is to improve and enhance the VQ code book. This program includes the simulation of BPNN training process on VQ of images and also includes the simulation of BPNN testing on VQ of images. Many gray level images with 256×256 dimension were used in simulation program as a training set for the BPNN training. Figure 2 shows examples of these images. Other images also with 256×256 dimension were used in program as a testing set. The implementation of BPNN VQ encoding can be summarized as follows:

Step 1: Divide the source image into non-overlapping blocks with a predetermined block dimension (P), where (P×P) equals the number of input layer neurons

Step 2: Take one block from the image, normalize it, rasterize it and then apply it to the input layer neurons of the BPNN

Step 3: Compute the output of hidden layer neurons by executing one iteration BP in a forward direction

Step 4: Open the codebook file to find the codeword that best matches the outputs of hidden layer neurons

Step 5: Store the index (position of this codeword in the codebook) in the compressed version of the source image file

Step 6: Repeat steps (2-5) for all blocks of the source image

The number of bits required for indexing each block equals $\log_2 M$ (where M is the codebook length).

At the beginning of the BPNN VQ compressed file, there is a header block that contains information about the: source image and compression process parameters.

The implementation of BPNN VQ decoding process can be described as follows:

Step 1: Open the compressed VQ (using BPNN) file

Step 2: Take one index from the compressed file

Step 3: Replace this index by its corresponding codeword obtained from the codebook. This codeword is assumed to be the output of hidden layer neurons

Step 4: Compute the output of the output layer neurons by executing one iteration BP in the forward phase. Then derasterize, denormalize and store this output vector in a decompressed image file

Step 5: Repeat steps (2-4) until the end of compressed file

EXPERIMENTAL RESULTS

The simulation of BPNN VQ image compression system (training and testing) is performed using the C# programming language. The compression system has been applied on well known images such as Hill, Vegetables and Lena images with 256×256 dimension as shown in Fig. 2 and deal with BMP graphic format images.

BPNN for reducing VQ codebook size: In this research, a BPNN is used for VQ compression of images. The BP algorithm is used to train this network to obtain the codebook with a smaller size than the codebook obtained by using Equitz Nearest Neighbour algorithm. The size of each codeword in the VQ BPNN codebook is equal to the number of hidden layer neurons. Whereas the size of each codeword in the codebook obtained using ENN algorithm

Fig. 2: Three samples of source and decompressed images (256×256) using BPNN VQ

is equal to P×P (where P is the dimension of the image block). For this reason, we get enhanced VQ codebook with less storage requirement.

Table 1 shows the VQ BPNN image compression system with a fixed size of image block dimension (P = 8), fixed number of input layer neurons (Ni = P×P = 64) and different number of hidden layer neurons (Nh). This table shows the effectiveness of the number of hidden layer neurons on the codeword size and codebook size. Images used are of size 256×256, so the resulted number of training patterns equals (((256×256)/(8×8)) = 1024). At the same time, the codebook size can be obtained from multiplying the number of training patterns by the codeword size.

Table 1: No. of hidden layer neurons vs. codebook size

VQ algorithm	Hidden layer neurons	Codeword size (in Bytes)	Codebook size (in Bytes)
VQ - ENN	-----	64	65536
VQ - BPNN	64	64	65536
VQ - BPNN	32	32	32768
VQ - BPNN	16	16	16384
VQ - BPNN	8	8	8192
VQ - BPNN	4	4	4096
VQ - BPNN	2	2	2048

BPNN generalization on reducing VQ codebook size: BPNN image compression system reduces the super codebook size. There is also another advantage of BPNN codebook that is the BPNN generalization (i.e., the ability of compressing untrained image). Generalization produces optimal codebook that is used for compressing untrained image that is not used during the codebook design. This

Table 2: BPNN codebook size vs. supper codebook size

No. of images	No. of codewords in supper codebook	No. of codewords in VQ BPNN codebook
1	16384	16384
2	2×16384	16384
3	3×16384	16384
4	4×16384	16384

Table 3: PSNR for various levels of noise in a digital/analog channel

Probability (P) of error	PSNR in VQ BPNN coding (dB)	PSNR in VQ coding only	Channel type
0	30.0	42.0	Digital
0.2	22.0	14.0	Digital
0	34.25	40.0	Analog
0.2	22.0	13.2	Analog

would result in producing a codebook of very smaller size than the super codebook. In super codebook, every image to be compressed must be trained first. Whereas in VQ BPNN, the codebook is designed using one or two images that are used later for compression. The difference between the VQ BPNN codebook size and the super codebook size can be shown in Table 2 when using one, two, three and four images. Each of these images is of dimension 256×256 and block dimension 2×2.

Error tolerant transmission using VQ BPNN: The VQ BPNN image compression system has the ability to reduce the errors that may occur during compressed image transmission through analog or digital channel. Let us take a source of Vegetables image shown in Fig. 2 with 256×256 dimension and 256 gray levels. Table 3 lists PSNR for decoded images when they are transmitted through a noisy digital or analog channel and that are encoded either using VQ BPNN or using VQ only.

DISCUSSION

In additional to provide the parallel computations that result in high computational rates, the used BPNN provides also the codebook with the same features of the super codebook designed by LBG algorithm (Nasrabadi *et al*., 1994). The BPNN codebook has smaller size in comparison with the super codebook size. The size of each codeword in the super codebook designed by LBG algorithm equals (P×P), where P is the image block dimensions. Whereas, the size of each codeword in BPNN codebook equals the number of hidden layer neurons and this number is always less than P×P. This would result in reducing the storage space required for codebook and also this would result in decreasing the computational load of the encoding and decoding processes.

Table 4 shows the Compression Ratio (CR) and PSNR related to applying many approaches of VQ image compression. We applied these approaches on 256×256 gray scale Lena image which segmented into blocks with 8×8 dimension. The CR is calculated using Eq. 2.

Table 4: CR and PSNR of different approaches of VQ image compression

Approaches	Compression rate (bpp)	PSNR (dB)
JPEG	25.04	25.30
Amerijckx *et al*. (2003)	25.22	24.70
Rawat and Meher (2009)	0.34	32.14
VQ coding only	0.30	42.00
Our approach (VQ BPNN)	0.15	30.00

From Table 4, we can note that our approach (VQ BPNN) got the best results related to CR.

As mentioned earlier in our previous research (Omaima, 2010), the performance of BP can be improved by applying many approaches as follows: adding bias unit; using momentum variable (θ); using beta; changing the learning rate value and introduction of random noise. In this research, we applied the same approaches to improve the BP in training the BPNN for VQ of images. We obtained best results by reducing the number of iterations in BP and reducing the size of VQ codebook.

Acceptability of image quality during the transmission can be achieved by using the minimum possible number of transmission bits. Transform coders are used for coding images at rates less than 1 bpp to produce more pleasing reconstructed images by reducing presence of channel errors. The small size of the codeword that is generated by VQ BPNN reduces the chance of error occurring during the codebook transmission.

Transmission process is simulated by adding random noise to the transmitted data (indexes of codewords). If the transmission channel is analog, noise is added to the value of codeword index so that it is deviated randomly by one level either upward or downward. But if the channel is digital, noise is added as suggested by Melesse and Hanley (2005) to one bit or more of the codeword index. The simulation of image transmission on noisy analog and digital channels was described in details in our previous research (Omaima, 2010).

CONCLUSIONS

A three layered VQ BPNN is used for building an improved codebook with smaller size than with the original VQ codebook to speed up VQ BPNN compression/decompression processes.

The BP was used for training the designed VQ BPNN. The BP algorithm used for VQ BPNN training is also called self-supervised because the output which corresponds to the certain input is not

provided by an external teacher. The BPNN is essentially an N to N encoder with analog inputs and outputs.

A simulation program for VQ BPNN image compression system (training and testing) was developed using C# language. A trained VQ BPNN was obtained to produce the codevectores of any image by the hidden layer neurons. We can then apply both trained and un-trained images to this VQ BPNN to compress them (but not in the same performance of the trained images) with no need to search for the closest codevector in codebook. This can be done especially when using small number of image block dimension (P) (i.e., small number of input layer units).

Experiments were conducted by changing VQ BPNN architecture. From the results, one can clearly see that the performance of the designed VQ BPNN image compression system can be increased by modifying the VQ BPNN architecture. This is done by: changing the number of input layer neurons and the number of hidden layer neurons. The performance can be increased also by: monitoring the total error value during the training process and changing the values of learning rate (η) and momentum variable (α) depending on this error; using the beta term (β) in the sigmoid function and finally by adding small random noise to BPNN weights to speed up this algorithm and also to enhance the VQ codebook.

Practically, we can note that the VQ BPNN has the ability to enhance any noisy compressed image that had been corrupted during compressed image transmission through a noisy digital or analog channel.

Finally, from the results, it is observed that proposed VQ BPNN is faster than other algorithms although it needs a time for learning process. At the same time, the codebook produced using VQ BPNN requires less storage than the super codebook produced by general VQ.

During the development of this study, recommendations for further work came to mind. The denormalization process (quantization process) that is applied on the hidden layer outputs during the compression process is from real format with range of (0-1) allocating four bytes for each pixel to integer format of range of (0-255) allocating one byte for each pixel. This quantization routine will perform a very simple equation to quantize the real format of the pixel value to the integer format which takes (8.bits). Instead of this, we can use more powerful quantization technique (Huffman Coding) which adds higher degree of compression.

REFERENCES

Al-Daoud, E., 2009. A comparision between three neural network models for classification problems. Int. Artif. Intell., 2: 56-64.

Amerijckx, C., J.D. Legat and M. Verleysen, 2003. Image compression using self-organizing maps. Syst. Anal. Modelling Simulat., 43: 1529-1543.

Bayazit, U. and W.A. Pearlman, 1999. Variable-length constrained-storage tree-structured vector quantization. IEEE Trans. Commun., 8: 321-331.

Bayir, R., 2008. Condition monitoring and fault diagnosis of serial wound starter motor with learning vector quantization network. J. Applied Sci., 8: 3148-3156.

Boopathy, G. and S. Arockiasamy, 2010. Implementation of vector quantization for image compression-a survey. Global J. Comput. Sci. Technol., 10: 22-28.

Gholizadeh, M.H. and M. Darand, 2009. Forecasting precipitation with artificial neural networks (case study: Tehran). J. Applied Sci., 9: 1786-1790.

Ibrahim, N.K., R.S.A.R. Abdullah and M.I. Saripan, 2005. Artificial neural network approach in radar target classification. J. Comput. Sci., 5: 23-32.

Jilani, S.A.K. and S.A. Sattar, 2010. JPEG image compression using FPGA with artificial neural networks. IACSIT Int. J. Eng. Technol., 2: 252-257.

Kohonen, T., 1990. The self-organizing map. Proc. IEEE., 78: 1464-1480.

Linde, Y., A. Buzo and R.M. Gray, 1980. An algorithm for vector quantizer design. IEEE Trans. Commun., 28: 84-95.

Melesse, A.M. and R.S. Hanley, 2005. Energy and carbon flux coupling: Multi-ecosystem comparisons using artificial neural network. Am. J. Applied Sci., 2: 491-495.

Narayanan, S. and S.S. Baboo, 2010. A novel enhanced neural network model for image compression using wavelet. MSLLFOCPN, Int. J. Comput. Applic., 6: 25-30.

Nasrabadi, N.M., C.Y. Choo and Y. Feng, 1994. Dynamic finite-state vector quantization of digital images. IEEE Transact. Comm., 42: 2145-2154.

Omaima, N.A., 2010. Improving the performance of backpropagation neural network algorithm for image compression/decompression system. J. Comput. Sci., 6: 1347-1354.

Ping, Y. and N.V. Anastasios, 1994. Hierarchical finite-state vector quantization for image coding. IEEE Trans. Commun., 42: 3020-3026.

Rawat, C.S.D. and S. Meher, 2009. A hybrid coding scheme combining SPIHT and SOFM based vector quantization for effectual image compression. Eur. J. Sci. Res., 38: 425-440.

Somasundaram, K. and S. Domnic, 2006. Modified vector quantization method for image compression. World Academy Sci. Eng. Technol., 19: 128-133.

Torres, L. and J. Huguet, 1994. An improvement on codebook search for vector quantization. IEEE Trans. Communicat., 42: 208-210.

Zhe-Ming, L. and L. Yue-Nan, 2010. Image compression based on mean value predictive vector quantization. J. Informat. Hiding Multimedia Signal Processing, 1: 172-178.