Near Lossless Compression of Video Frames Using Soft Computing Technologies in Immersive Multimedia

Prakash Jadhav, G. K. Siddesh

Abstract*—Compression of Video Frames is a heavily researched domain in Digital TV Technology and Multimedia applications and hundreds of researchers have experimented with various techniques of transform coding and quantization mechanisms with varied results. Despite this fact, there is still a tremendous amount of scope for better compression schemes that seek to reduce the utilized band of the spectrum. In real time applications such as Digital TV and streaming Multimedia applications, time plays a crucial role and a successful implementation usually makes a compromise between factors such as (1) quality of reconstructed frames, (2) amount of utilized band owing to compression and (3) ability of the system to cope with increasing frame rates and advanced video profiles that are creeping into Digital TV Standards. It is important to observe that methodologies employed in achieving successful compression based on both inter and intra-frame redundancies rely very heavily on the frequency-domain redundancies generated by transform coding techniques and it is imperative that compression techniques place a great amount of emphasis on the correlation that can generated on a sub-image basis. Noise in images tends to decrease the correlation and noisy frames tend to compress less. Our objective in this research program is to focus on ways and means of increasing the redundancies by employing Linear Neural Network techniques with feed-forward mechanisms. Since quantization plays a major role in the amount of compression generated and the quality of the reconstructed frames, adaptive quantization based on a PSNR threshold value is employed so that best compression with a guaranteed reconstructed quality is obtained. Our research program targets to achieve the objectives: (1) better compression ratios with even fairly noisy frames (2) better quality of reconstructed frames in terms of very large PSNR and RMS Error tending towards values extremely small and nearer to 0 and (3) honoring the time constraints imposed by frame rates.

***Keywords:-** Neural Networks, Peak Signal to Noise Ratio (PSNR), Root Mean Square Error (RMS), Quantization, Discrete Cosine Transform (DCT), Burrows Wheeler Transform (BWT), Rosseta Vector, bandwidth.*

I. INTRODUCTION

Virtual Reality is a reality these days. What was once considered as an esoteric or arcane technology is a robust area of intensive research today, giving an added dimension to video, audio and environment. This so-called "immersive multimedia" immerses the subject in real-life environment of audio, video and other environmental scenarios making the subject feel that he is living in the real world with interactive environment surrounding him. Real-life experience is the basic theme of Immersive Multimedia. In the following paper we seek research on one aspect of Virtual Reality i.e. Video. Since streaming video frames tend to occupy a large amount of memory space in transporting over a network, compression is absolutely essential in reducing bandwidth required. The essence of this paper is in devising a method of compression that takes advantage of neighborhood redundancy in performing a near-optimal compression using soft computing methods.

II. LITERATURE SURVEY

The existing implementations of Compressors and Decompresses (**CODEC** for short) follow (and expected to follow) and adhere to ISO Standard 13818. The standards have evolved from MPEG-1 to MPEG-2, MPEG-4, MPEG-7 and MPEG-21. MPEG2 implements the CODEC based on DCT. There are implementations of CODEC based on H.264 Standard which uses Wavelets instead of DCT. The shortcomings of existing CODECs are: they are not adaptive to the patterns of the pixel population in the video frames. MPEG2 uses a flat and uniform quantization while other implementation uses Vector Quantization techniques. It is important to observe here that Vector Quantization, while being marginally superior in terms of compression ratio, has substantial computing overheads which offset the gain resulting from better compression ratio for a given visual quality metric. There are however various esoteric mechanisms proposed by researchers but none of them are sufficiently suitable for a straightforward and efficient implementation. When we consider the process of decoding pictures at the receiver end, every millisecond of time gained in the decoding process leads to better bandwidth utilization and overall efficiency. There is therefore a genuine need for better mechanisms of compression which will seek to achieve (1) better compression ratio (2) faster decoding of pictures at the receiver end and (3) better quality of reconstructed frames for a given quality factor in terms of Mean Square Error and Peak Signal to Noise Ratio..

III. PROPOSED FRAMEWORK

The proposed framework is based on adopting an adaptive quantization technique and deploying a linear Neural Network in order to perform the

Manuscript Received on November 2014

Prof. Prakash Jadhav, Electronics and communication Department, Visveswaraya Technological University, Belgaum/ K. S. School of Engineering and Management / Bangalore, India.

Dr. G. K. Siddesh, Electronics and communication Department, Bangalore University/ J.S.S. Academy of Technical Education / Bangalore, India.

compression. It is a well known fact that quantization controls the amount of information that is being discarded as being non-contributive to the human visual system. A flat quantization as adopted by MPEG2 CODEC does not take into account the significance of the transform coefficients. Owing to vast structural differences in varied types of images, a flat quantizer cannot discriminate between contributing and minimal-contributing coefficients. This being the case, sections of images with excessive details (with significant coefficients spread across the frequency matrix) may lose useful information in the quantization stage and compression may not be the maximum. Therefore, an adaptive quantizer that suits itself to the local characteristics of the blocks performs best.

In this research work, we deploy a linear Artificial Neural Network to generate an adaptive quantizer which runs through limited and predictable iteration cycles to evolve a suitable matrix. The proposed transform coding is the conventional 8x8 DCT, since DCT has very high energy-concentrating properties. Other transforms such as Wavelets, Walsh-Hadamard are also possible. The proposed methodology consists of the following steps:

i. Divide the image into blocks of 8x8
ii. For each block the following set of actions are taken in the specified order:
iii. The input to the input layer is an 8x8 block (64) pixels
iv. DCT is performed on the block, generating an 8x8 Frequency Matrix
v. The default quantization matrix is applied
vi. The quantized coefficients are reordered using Burrows-Wheeler Transform
vii. Huffman Coding is employed to compress the block and the compressed block is stored
viii. Inverse Huffman Coding is used to decompress the compressed output of the previous stage
ix. Inverse Burrows-Wheeler Transform is applied to the output of the above step
x. The inverse quantization is applied to regenerate the transform coefficients
xi. Inverse DCT is performed to reconstruct the 8x8 sub-image and stored
xii. The PSNR Value is calculated on the sub-images of steps 7 and 11
xiii. Adjust the values in the quantization matrix (Weight Adjustment)
xiv. Go to step 3 if PSNR does not meet the prescribed threshold

At first sight, it appears that the computational steps involved are too high and time consuming but in reality, the situation is different. Most of the modern day processors support threading concept so that several activities (that are mutually exclusive) can run in parallel. Practical implementations of CODEC intended for commercial use employ Digital Signal Processors with pipelines that run in parallel. Even the most modest DSP supports pipelines and parallel processing. Floating point DSPs can execute floating point instructions in a couple of processor clocks. This being the case, we will be justified in assuming that the apparently long sequences of processing steps mentioned above are of no consequence. The following pages describe the researched work in detail under the following various sections:

- Brief Exposition of lossy compression based on transform-coding (specifically DCT)
- Burrows-Wheeler Transform and Adaptive Quantization
- Structure of the proposed Neural Network
- Implementation of the Neural Network
- Results obtained and comparison charts

IV. LOSSY COMPRESSION OF VIDEO FRAMES AND TRANSFORM CODING

Compression of Video Data is a very important aspect of Digital TV transmission and reception. It is by no means an exaggeration to state that Digital TV technology wouldn't exist if compression of images weren't possible. It also implies that bandwidth reduction wouldn't be possible without the employment of compression technology in Digital TV Video Streams. In this session, we will try to understand in depth, the various techniques employed in performing lossy compression of video frames in Digital TV.

A. What does Lossy Compression achieve?

With lossy compression, we achieve a reduction in data volume, i.e. the compressed data requires much less space for storage. This in turn implies that the amount of information transmitted per second is less. Digital TV frames are streaming in nature and the frames are displayed on the display either at 30 or sometimes 60 frames/ second depending on the MPEG Profile adopted. This means that a tremendous amount of data has to be transmitted per second and reducing this amount will definitely ease the efforts required by the transmitting equipment. Also, a substantial amount of time saved in transmitting the compressed data can be utilized by the transmitting equipment for other useful and interesting activities such as providing user-interaction in Interactive TVs. We will elaborate the above situation by introducing the concept of bandwidth. In Digital TV terms, a bandwidth can be loosely defined as the number of bits required to be transmitted per second. The following table illustrates the bandwidth requirements for various types of Digital TV display formats:

Table 1: Digital TV display formats

1	2	3	4 = [2] x [3]	5 = [4] x 24	6	7 = [5] x [6]
No:	Display Width Pixels	Display Height Pixels	Total Pixels	Total Number of bits at 24 bits/pixel	Frame Rate	Number of Bits to be transmitted per second
1	352	288	101376	2433024	30	72990720
2	720	576	414720	9953280	30	298598400
3	720	608	437760	10506240	30	315187200
4	960	576	552960	13271040	30	398131200
5	1440	1152	1658880	39813120	60	2388787200
6	1920	1152	2211840	53084160	60	3185049600

From the above table, it is obvious that as the display size increases or the frame rate increases the total number of bits to be transmitted per second increases dramatically. From 1 (low profile display) to 6 (high definition profile), the values in the last column vary between 72 million to 3 billion bits per second. Even for a low profile display, 72 million bits per second is no small a figure. It is

therefore important to compress the data before transmission. Compression by even about 50% (can be easily achieved) can exhibit dramatic improvement in performance. The following pages elaborate on the theme of lossy compression.

B. Compression Fundamentals:

Data Compression broadly falls into two categories – lossless and lossy. A compression scenario involves three distinct entities:

1. Data to be compressed
   Size: $S_{original}$ Data: $D_{original}$
2. Compressed Data
   Size: $S_{compressed}$ Data: $D_{compressed}$
3. Decompressed Data
   Size: $S_{decompressed}$ Data: $D_{decompressed}$

A compression is supposed to have taken place if:

$$S_{compressed} < S_{original}$$

The compression is lossless if:

$$D_{original} \equiv D_{decompressed}$$

The compression is lossy if:

$$D_{original} \not\equiv D_{decompressed}$$

These axioms are simple enough and don't need any explanation. Lossless compression is required for compression of data files containing numeric values, executable/ DLL files containing code and text files containing readable information. Even a slight loss of information will render data useless for consumption. Popular compression and archival programs like *WinZip, WinRar, Gzip* etc. employ lossless compression. There are certain types of data such as video and audio signals that can be compressed with some amount of loss and yet we may be able to retain the substance and detail of the signal with reasonable fidelity. This is certainly true of Images and streaming video where we can afford to discard a substantial amount of chromatic information by the process of sub-sampling and yet retain the perceptual quality to an acceptable level. Extensive experiments in color science have proven the fact the human eye is less sensitive to chroma than luma. This fact coupled with spatial redundancy and temporal redundancy (in streaming video) contributes to a large measure in achieving surprising compression ratios. The spatial redundancy manifests itself much better in the transform domain (either DCT or wavelets) and judicious discarding of transform coefficients that are not significantly contributing to the image detail by quantization techniques. Thus, transform coding and quantization play a major role in making image compression attractive. The following paragraphs seek to explain very briefly these two concepts.

C. 2-Dimensional Discrete Cosine Transform

The Discrete Cosine Transform and its inverse for 2D Images s defined by the equations:

$$F(u,v) = \frac{2}{\sqrt{WH}} C_u C_v \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} p(x,y) * \cos\frac{2x+1}{2W}\pi u * \cos\frac{2y+1}{2H}\pi v \quad \textit{Forward DCT}$$

$$p(x,y) = \frac{2}{\sqrt{WH}} \sum_{u=0}^{W-1} \sum_{v=0}^{H-1} C_u C_v * F(u,v) * \cos\frac{2x+1}{2W}\pi u * \cos\frac{2y+1}{2H}\pi v \quad \textit{Inverse DCT}$$

where p(x,y) is the pixel at spatial coordinate (x,y)
F(u,v) is the frequency coefficient at coordinate (u,v)
W is the width of the subimage and H its height

The invention of this transform is attributed to **Ahmed** and **Natarajan** [N. Ahmed, T. Natarajan and K. R. Rao, "Discrete cosine transform", *IEEE Transactions onComputers*, Vol. C-23, 1974, pp. 90–93.]

This transform is remarkable in several ways – specifically for its energy concentrating properties in the lower order coefficients, generating a large amount of sparseness through quantization. DCT is the corner stone of innumerous compression algorithms and commercial compression programs for video data. It is interesting to note that while DCT is used extensively in lossy compression mechanisms, DCT by itself is perfectly reversible with absolutely no loss except for floating point imprecision and rounding errors which are usually negligible. The real loss in lossy compression comes from the quantization stage where integer division truncates the fractional part of the coefficients.

V. BURROWS WHEELER TRANSFORM (BWT)

In transform coding, one usually rearranges the transform coefficients in the increasing order of frequency coordinates, the premise being that as the frequency coordinate increases, the transform coefficients tend to decrease, losing significance. With such rearrangement, one usually finds the coefficient values monotonically decreasing in the increasing coordinate direction. This implies an increased level of correlation, leading to better compression. But in certain images with unusual structural features, this is far from truth.

Sorting of coefficients in the descending order will definitely enhance the correlation in all the cases (including extreme cases), but sorting unfortunately is not reversible. Here, Burrows Wheeler Transform comes to our rescue. It is a simple and wonderful algorithm that will preprocess the DCT coefficients to achieved substantially enhanced levels of compression.

A. The Burrows-Wheeler Transform Basics

Michael Burrows and David Wheeler released a research report in 1994 discussing work they had been doing at the Digital Systems Research Center in Palo Alto, California. Their paper, "A Block-sorting Lossless Data Compression Algorithm" presented a data compression algorithm based on a previously unpublished transformation discovered by Wheeler in 1983.

A block of data that is the input to BWT algorithm is rearranged using a sorting algorithm. The output of BWT is a block that contains exactly the same original data that was input, differing only in their ordering. The transformation is reversible in the sense that the original ordering of the data elements can be restored without loss of information integrity.

The BWT is performed on an block of data at once. Commonly known lossless compression algorithms operate in the streaming mode, at a time it reads a single byte or a few bytes. With BWT one can operate on large blocks of data.

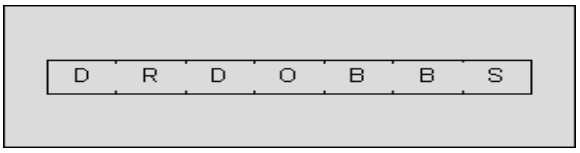


Figure 1: An sample data set

To illustrate the workings of BWT, consider the string shown in figure 1 which has 7 bytes of data. We start with the original string and build more strings by rotating one character to the right at a time. So, we end up with 7 strings (including the original). This is depicted in figure 2 below.

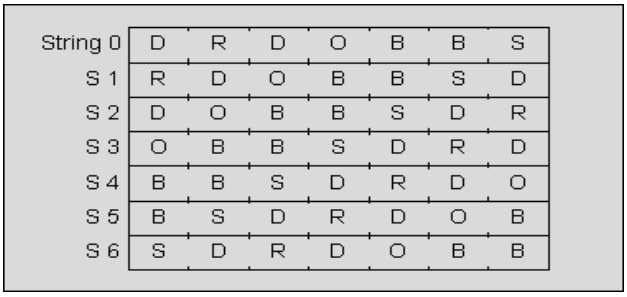


Figure 2: Set of strings associated with the buffer

The next step is to perform a lexicographical sorting of inputs (usually a *Quick Sort implementation*). There are two important points to note in this picture. First, the strings have been sorted, but we've kept track of which string occupied which position in the original set. So, we know that the String 0, the original unsorted string, has now moved down to row 4 in the array.

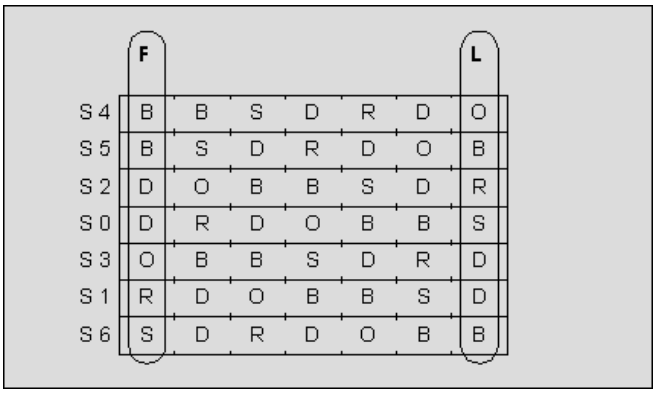


Figure 3: The set of strings after sorting

We tag the first and last columns with special designations F and L for the first and last columns of the array block. Column F contains all the characters in the original string in sorted order. So the original string "DRDOBBS" is represented in F as "BBDDORS". The characters in column L don't follow any ordering, but interestingly they have a notable property. Each of the characters in L is the *prefix character* to the string that starts in the same row in column F. The actual output of the BWT, consists of two things: a copy of column L, and the *primary index*, an integer indicating which row contains the original first character of the buffer B. So performing the BWT on our original string generates the output string L which contains "OBRSDDB", and a primary index of 5.

The primary index 5 can be ascertained easily since the original first character of the buffer will always be found in column L in the row that contains S1. Since S1 is simply S0 rotated left by a single character position, the very first character of the buffer is rotated into the last column of the matrix. Therefore, locating S1 is equivalent to locating the buffer's first character position in L.

At first sight the above transform doesn't seem reversible. Generally sorting routines are not reversible, so a reversal of Quick Sort doesn't exist – you will never be able to get the original ordering by any means. But in the case of BWT, a reversal is indeed possible through a transformation vector. The transformation vector is an array that defines the order in which the rotated strings are scattered throughout the rows of the matrix of Figure 3.

The transformation vector, *T*, is an array with one index for each row in column F. For a given row *i*, T[i] is defined as the row where S[i + 1] is found. In Figure 3, row 3 contains S0, the original input string, and row 5 contains S1, the string rotated one character to the left. Thus, T[3] contains the value 5. S2 is found in row 2, so T[5] contains a 2. For this particular matrix, the transformation vector can be calculated to be { 1, 6, 4, 5, 0, 2, 3 }.

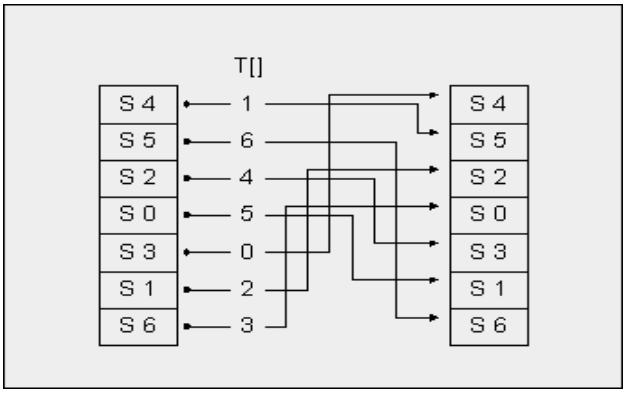


Figure 4: The transformation vector routes S[i] to S[i + 1]

Figure 4 shows how the transformation vector is used to walk through the various rows. For any row that contains S[i], the vector provides the value of the row where S[i + 1] is found. This transformation vector is called '*Rosetta Vector*'. Therefore, given a copy of L, we can calculate the transformation vector for the original input matrix. And consequently, given the primary index, you can recreate S0, or the input string. The possibility of calculating the transformation vector requires only that we know the contents of the first and last columns of the matrix. This is simply having a copy of L.

B. Adaptive Quantization Mechanism

1	1	1	2	3	4	5	6

1	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8
2	3	4	5	6	7	8	9
3	4	5	6	7	8	9	10
4	5	6	7	8	9	10	11
5	6	7	8	9	10	11	12
6	7	8	9	10	11	12	13

Figure 5: Quantization Matrix (Level 0)

The contents of the white cell (with 1) will remain unchanged at all quantization levels. All the values in the gray colored cell with increase by 1 (stepping size) at each increase in quantization level. The columns and rows are reckoned with starting value 0, so that rows (**R**) have the range 0 → 7 and cols (**C**) have range 0 → 7. So, given **R** and **C,** we can calculate the cell value using the following equation:
Quantization Coefficient for level 0:

$$(R<3)*((C<(3-R))*1+(C>(2-R))*(C+R-1))+(R>2)*(R+C-1)$$

Important Note:
Here, we would like to introduce a C Language like convention concerning relational operators: $<, >, \leq, \geq, ==$. The operator '==' (double equal sign) denotes equality, i.e. the left and right sides of the operator are equal. The result of a logical expression such as $R < 3$ is either true (1) or false (0). The interpretation of the above equation is based on this convention.

The logic of keeping the values in the white cells unchanged is fairly straightforward. The coefficients at the lower frequency coordiante end are contributing terms and the 6 coefficients in these positions are therefore significant in almost all the cases.

In implementing this quantizer, we employ a increment step size of 1, though larger values are possible. Larger step sizes lead to coarser and more approximated images with smaller values of PSNR and this is not preferable, as our objective is to achieve two distinct but conflicting goals – *better decompressed quality* and *larger compression ratios*.

C. Operation of the Neural Network

The proposed Neural Network is a linear feed forward network. The weights to the input layer are fixed by the quantization matrix shown in the earlier pages. The network consists of 64 inputs (the pixels of an 8x8 sub-image). Unlike the conventional ANN, the iteration process goes through 10 stages in parallel. These stages are designated as Levels 0 through 9. Level 0 receives the quantization matrix shown above. Subsequent levels use quantization equation:

$$(R<3)*\Big((C<(3-R))*1+(C>(2-R))*(C+R-1)\Big)+(R>2)*(R+C-1)+Level$$

This means in simple terms that the values in the gray area of the quantization matrix are increased by 1 (stepping size) at each level. Level 0 will generate the best quality and Level 9 will generate a relatively poor quality of reconstructed image. More levels are unnecessary as extremely coarse quantization will generate frames of unacceptable quality.
Given a sub-image $S(x,y)$ Quantization Matrix at Level L $Q_L(x,y)$ and an output matrix $O(x,y)$, quantization is performed as per equation: $O(x,y) = S(x,y)/Q_L(x,y)$, the division being on a term by term basis.
The given image is divided into blocks of size 8x8 and the same sub-image is fed into all the 10 levels. The Neural Net performs the 8x8 DCT followed by quantization, reordering using Burrows Wheeler Transform and compression using Huffman coding. The outputs at each level are: (1) compressed sub-image and (2) PSNR value based on the input $PSNR_k$ (PSNR at Level k). The Peak-to-Signal-Noise-Ratio is defined as:

$$MSE = \frac{1}{WH}\sum_{x=1}^{W}\sum_{y=1}^{H}[g(x,y) - g'(x,y)]^2$$

where, W is the width of the image and H is the height of the image.

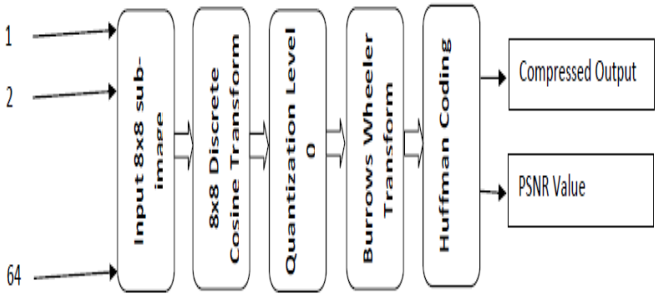
$$PSNR = 20.0 * log_{10}\frac{L}{\sqrt{MSE}}$$

Each Level generates a compressed sub-image with a corresponding PSNR. The input to the Neural Net at the start of compression is a threshold value for the PSNR with a tolerance band of 100 dB. So, given a threshold value $PSNR_T$, it is a matter of deciding while level to go to for picking up the compressed sub-image. So, the acceptable range of PSNR should satisfy the condition (assuming the value at level L is $PSNR_L$):

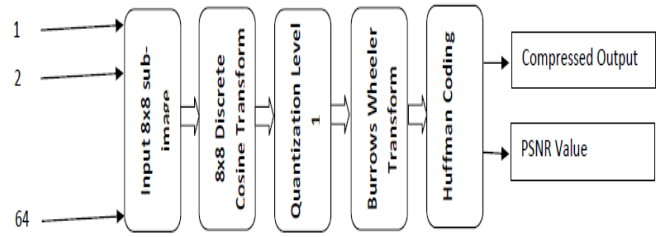
$$(PSNR_T - \delta) < PSNR_L < (PSNR_T + \delta) \text{ where } \delta \text{ is the tolerance band}$$

We perform this exercise bottom-up starting at Level 9 and proceeding to Level 0 one at a time, applying the above condition. In the climbing up process, the compressed sub-image that matches the above relation earliest is output as the compressed sub-image. Additional 1-byte information regarding the level that generated this output is also emitted. This information is necessary for the decoder to decompress the sub-image successfully. The above process is repeated until all the sub-images are processed and compressed.

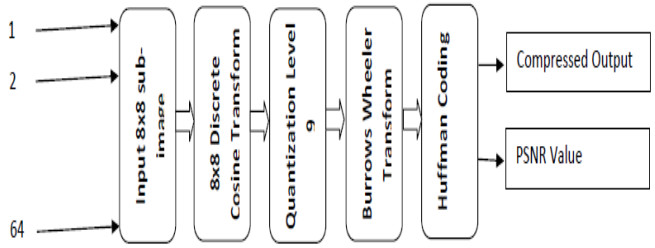
Layout of the Linear Neural Network



Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.



Intermediate Levels 2 to 8



A few important observations are in place. Unlike the traditional ANN, training sets and training are not necessary. The number of iterations for converging to a solution is finite and limited to 10 only. Because

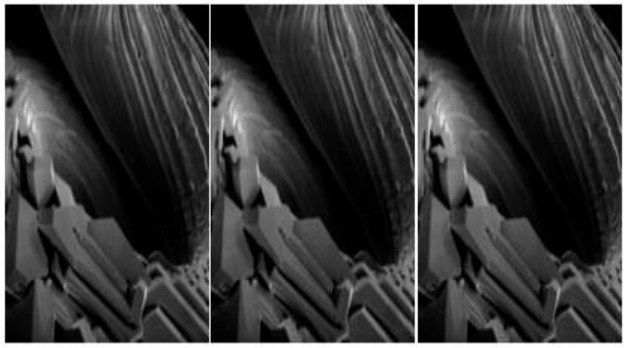
The decoding process is exactly the inverse process of encoding.

i. Determine the quantization level from Block Header and generate the quantization matrix
ii. Use Inverse Huffman Coding to decompress the compressed output
iii. Use Inverse Burrows-Wheeler Transform to the output of the above step
iv. Apply inverse quantization to regenerate the transform coefficients
v. Apply Inverse DCT to reconstruct the image and store it

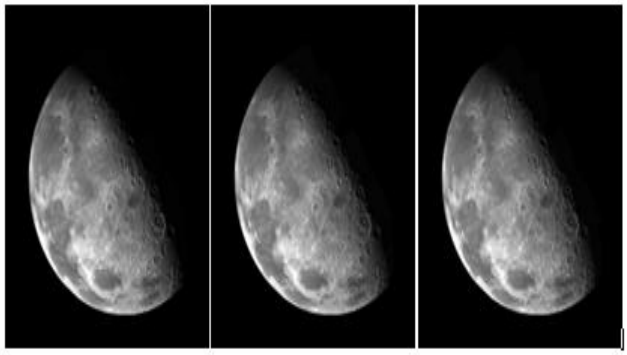
VI. Results

Results of the Network with Comparisons (Ours vs. MPEG2) PSNR threshold at 4500 dB

Original Image 610 KB ANN Compression 61 KB MPEG2 Compression 124 KB



Original Image 482 KB ANN Compression 41 KB MPEG2 Compression 102 KB

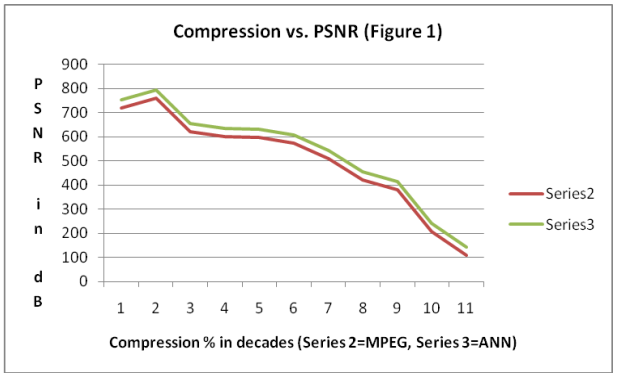


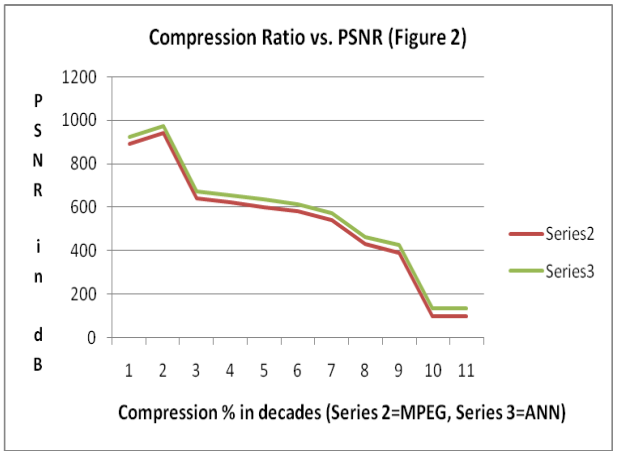
Original Image 398 KB ANN Compression 21 KB MPEG2 Compression 97 KB

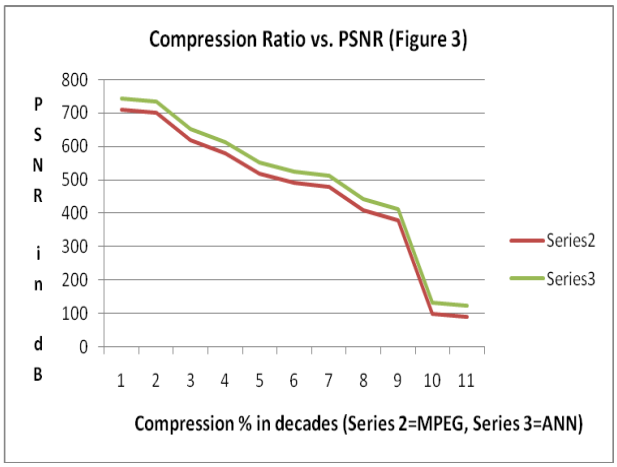
levels, comp

Graphical Results:

Compression % versus PSNR in decibels







The results of compression based on PSNR based on bit rate

Examination of the graphs show that ANN compression produces larger PSNR values, thereby proving that fidelity of decompressed image is more in the case of ANN compression as compared to MPEG2 compression.

For building ANN and introducing the soft-computing components, MATLAB has been used. Since most of the components used in this research are not built-in functions in MATLAB, scripts were written to perform these operations specifically, quantization procedures, specialized Neural Network routines compression methods based on modified cosine transforms etc., Burrows-Wheeler transforms etc.

VII. CONCLUSION

On all the image files without exception, ANN compression produced better compression ratios for a given quality factor. For evaluation of performance in compression time, we performed the compression of several images both by ANN and MPEG2 Codec 500 times in a loop to determine the average time taken. While time for compression understandably varies across different images, the performance of ANN was much better than that of MPEG2 video codec. The algorithm presented here lends itself very easily for implementation both in hardware and software. Future enhancements and extensions to this research work could include deployment of an additional layer clustering based on Markov's Chains. The structure of the implementation is such that any type of transform can be employed in the Neural Network, potential candidates being *wavelets* (in all its variations), *Hadamard* and *Walsh Transforms*.

REFERENCES

[1] Ko Nishino, Yoichi Sato and Katsushi Ikeuchi, "Eigen-Texture Method: Appearance Compression and Synthesis Based on a 3D Model" IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 23, No. 11, pp 1257-1265, November 2010
[2] Sang-Uok Kum, Ketan Mayer-Patel, Henry Fuchs, "Real-Time Compression for Dynamic 3D Environments" ACM, MM'03, pp – 2-8, November 2003
[3] G. Nur Yilmaz, H.K. Arachchi, S. Dogan, A. Kondoz "3D video bit rate adaptation decision taking using ambient illumination context" Engineering Science and Technology, an International Journal 17 pp 105-115 may 2014
[4] Jacky C.P. Chan, Howard Leung, Jeff K.T. Tang, and Taku Komura, "A Virtual Reality Dance Training System Using Motion Capture Technology", IEEE Transactions on Learning Technologies, Vol. 4, No. 2, pp- 187-195, April-June 2011
[5] G Boopathi, Dr.S.Arockiasamy, "An Image Compression Approach using Wavelet Transform and Modified Self Organizing Map" IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 5, No 2, pp- 323-330, September 2011
[6] Dong Zhang, Bin Li, Houqiang Li, "Intermedia-Based Video Adaptation System: Design and Implementation", TSINGHUA SCIENCE AND TECHNOLOGY, Volume 17, Number 2, pp113-127, April 2012
[7] Chinnusamy.K, Abinaya Preethi.K," Performance and Analysis of Video Streaming of Signals in Wireless Network Transmission", IOSR Journal of Electronics and Communication Engineering (IOSR-JECE), Volume 9, Issue 1, Ver. I PP 11-15 Jan. 2014,
[8] Gary J. Sullivan, Jens-Rainer Ohm et,al. "Overview of the High Efficiency Video Coding (HEVC) Standard" IEEE Transactions On Circuits And Systems For Video Technology, Vol. 22, No. 12, Pp 1647-1668, December 2012.
[9] Gulistan Raja1and Muhammad Javed Mirza, "In-loop Deblocking Filter for H.264/AVC Video" Advanced Engineering Research Organization, Wah Cantt.
[10] Mahsa T. Pourazad, Colin Doutre, Maryam Azimi, And Panos Nasiopoulos, "HEVC: The New Gold Standard For Video Compression" IEEE Consumer Electronics Magazine , *Digital Object Identifier 10.1109 / Mce.2012.2192754, pp 36-46 22 June 2012*

Prof. Prakash Jadhav is an Assistant professor in K. S. School of Engineering and Management Bangalore, received his B.E. degree in S.T.J.Institute of Technology. Ranebennur in 1999 and M.Tech degree in A.M.C. Engineering College, Bangalore, in 2006, and pursuing Ph.D in Visvesvaraya Technological University Belgaum, Karnataka, India, He has two international conference publications to his credit. Research work is carried out in Multimedia Networking and Communication, and has IEEE & MISTE membership, achieved best paper presentation in "KNOWLEDGE UTSAV" conducted in S.B.M.J.C.E., Bangalore.

Dr. Siddesh. G.K. is an Associate Professor in JSS Academy of Technical Education Bangalore, obtained his Bachelors and Masters Degree in Electronics and Communication Engineering from Bangalore University and Manipal Academy of Higher Education , Karnataka respectively. He also obtained his doctrol degree from Visvesvaraya Technological University Belgaum, Karnataka, India. He has 5 international journal publications and one international conference publication to his credit. His research area and area of specialization includes Wireless communication, Computer Networks, Image and Data compression and Processing.

.