LOSSLESS VERSUS LOSSY COMPRESSION: THE TRADE OFFS

[1]**Ayush Raniwala**
Student, Computer Science Dept.
SVKM's NMIMS Mukesh Patel School of Technology Management & Engineering,Mumbai, India

[2]**Shreyas Singhvi**
Student, Computer Science Dept.
SVKM's NMIMS Mukesh Patel School of Technology Management & Engineering,Mumbai, India

ABSTRACT—This paper discusses lossless and lossy data compression in text and image compression and compares their performance in the same. The paper begins with selecting the best possible basic lossless compression technique from the one chosen and comparing it with another basic lossy technique. The techniques considered are Huffman, Shannon–Fano Coding and SPIHT techniques. So far the preferred technique for image compression has been lossy since the performance is found to be far superior to that of lossless but using this can sometimes lead to loss of important information, thus this paper aims to help in selecting the best possible algorithm depending on the task in hand.

Keywords— Image Compression, Lossless Compression, Lossy Compression, Huffman Coding, Shanon-Fano Coding, SPIHT based image compression

I. INTRODUCTION

Compression is the procedure of encoding information to less bits than the first representation so it consumes less storage space and less transmission time while conveying more than a system [1].

Compression can be classified into two types lossless and lossy compression, the two differ on the basis of how the data recovered is compared to its initial form, As the name suggests in lossless compression techniques, no information is lost. In other words, the reconstructed data from the compressed data is identical to the original form. Whereas in lossy compression, some information is lost, i.e. the decompressed data is similar to the original but not identical. Data compression is possible due to the redundancy in the everyday data.

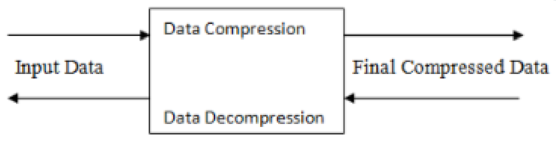

Fig 1. Data Compression and Decompression [2]

Uncompressed multimedia (graphics, audio and video) data requires considerable storage capacity and transmission bandwidth. Despite rapid progress in mass-storage density, processor speeds, and digital communication system performance, demand for data storage capacity and data-transmission bandwidth continues to outstrip the capabilities of available technologies. The recent growth of data intensive multimedia-based web applications has not only sustained the need for more efficient ways to encode signals and images but have made compression of such signals central to storage and communication technology. With this is in mind it has become very important for us to find the perfect technique for image compression since in certain application's loss of even a bit of data is not tolerable but at the same time needs to be quickly transmitted an example of this would be an x-ray of a patient sent to his doctor, whereas in some minimal loss of data wouldn't matter if the rate of compression is higher.

II. LOSSLESS COMPRESSION

Lossless data compression is a procedure that permits the utilization of data compression calculations to pack the content data furthermore permits the precise unique data to be remade from the compacted data as shown in Fig 1.

The prevalent ZIP record organize for data compression is a use of lossless data compression approach. Lossless compression is utilized when it is vital that the first data and the decompressed data be indistinguishable. Most lossless techniques utilise the redundancy in data by combining them together during compression, During image compression, compression is achieved by removing unnecessary metadata from JPEG and PNG files. RAW, BMP, GIF, and PNG are all lossless image formats. The only disadvantage of lossless image compression is that the size of the file is comparatively higher then what it would be with lossy.

Lossless compression techniques may be classified by kind of data they are intended to pack. Most lossless compression projects utilize two various types of calculations: one which creates a factual model for the info data and another which maps the information data to bit strings utilizing this model as a part of such a route, to the point that as often as possible experienced data will deliver shorter yield than improbable data [2].

A. *Basic Lossless Techniques*

The techniques briefly studied are:

1. Huffman coding
2. Shannon–Fano Coding

1. Huffman Coding:

Huffman data compression is a lossless compression technique which use's the probability of the occurrence of a symbol to remove redundant data, it follows top down approach where the binary tree is built from the top down to generate an optimal result, the characters in a data file are converted to binary code where the most common characters in the file have the shortest binary codes and the least common characters have the longest binary code. A Huffman code can be determined by successively constructing a binary tree as shown in Fig 3, whereby the leaves represent the characters that are to be encoded, every node contains the probability of occurrence of the characters belonging to the sub tree beneath the node, the edges are labelled with the bits 0 and 1. It is also known as prefix coding or prefix elimination coding. Illustrated below in Fig 2, is the method to find the codes of the respective entities after the binary tree is formed.

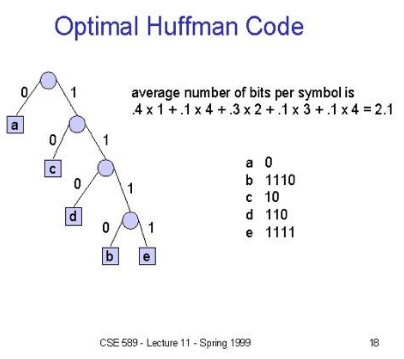
Optimal Huffman Code
average number of bits per symbol is
.4 x 1 + .1 x 4 + .3 x 2 + .1 x 3 + .1 x 4 = 2.1
a 0
b 1110
c 10
d 110
e 1111
CSE 589 - Lecture 11 - Spring 1999
18

Fig 2. Huffman Coding

COMPRESSION ALGORITHM

The following are the steps for compression:

1. Calculate frequency of occurrence for every character which is done by two methods one in which fixed look up tables are directly provided at encoder and decoder side this is known as static Huffman encoding and the other in which a running estimate of input letter frequency is maintained this is known as dynamic Huffman encoding.
2. The 2nd step is to form a Huffman code tree, starting with the leaf nodes. The method goes on replacing two minimum frequency nodes with a new parental node, and assigns the sum of the frequency to it.
3. In the 3rd step bit '0' or '1' are assigned to the branches of the code tree, with left/upper going branches as 0 and right/lower going as 1
4. The final step is to trace down the generated Huffman codes for every symbol, starting from the root node.

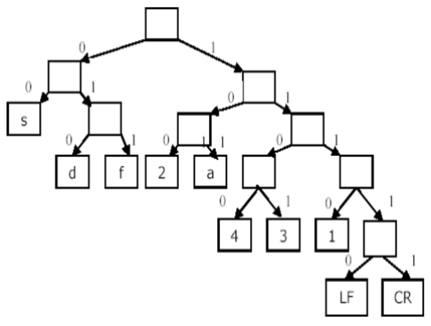

Fig 3. Huffman Coding Tree

2. Shannon–Fano Coding:

Shannon-Fano coding provides a similar result compared with Huffman coding at the best. It will never exceed Huffman coding.

The Shannon-Fano algorithm was the very first method in the data compression sphere hence it is not optimal but it sets a precedent for many forthcoming methods.

Description of algorithm:

While defining the method of compression Claude Shannon defined a term called entropy, adopted from thermodynamics.

There are three main rules:

1. Every code has varying bits.
2. Code: Encoding: High probability- lower bits, Low probability- higher bits.
3. Varying length of code does not affect decompression algorithm.

COMPRESSION ALGORITHM

The algorithm is divided into three main stages which are as follows:

1. First Stage: Correspondence table generated after completely parsing data file. The correspondence table consists of 3 things: frequency of symbol, code of symbol & ASCII code of symbol.
   Correspondence table: Arranged in descending order of frequency of occurrences.
2. Second Stage: all symbols are uniquely codified using an elaborate recursive function.
   Current table: divided into 2 sub tables having approximately equal cumulative frequencies. A '0' is added before the upper table and a '1' is added before the lower one. This is process keeps having until each table consists of a single element.
3. Third Stage: the codes are used to translate the original data to a compressed format. Table has to be arranged in order of their ASCII codes tp perform this action.

The correspondence sequence appears as a string of bytes.

DECOMPRESSION ALGORITHM

The following are the steps for decompression:

1. Create decompression tree.
2. Use tree to decompress file.
3. The file is read bit by bit
4. Root is used as starting point
5. If the bit is: '- 1' - there will be a right shifting, if there is a node -the next bit is read; '0' - there will be a left shifting, if there is a node -the next bit is read;
6. If there isn't any node – the character is written as a leaf; - the next bit is read; - we start back from the root;

B. Modification on the techniques

1. **Huffman Coding (Modified):**

The proposed technique for modification is the dynamic bit reduction technique which was found to have a better result then both bit reduction and basic Huffman technique's.

The algorithm works in three phases to compress the text data. In the first phase data is compressed using dynamic bit reduction technique, in the second phase unique words are found to compress the data further and in the third and final phase Huffman coding is used to compress the data in a random data set as shown in fig 4, further to produce the final output [3].

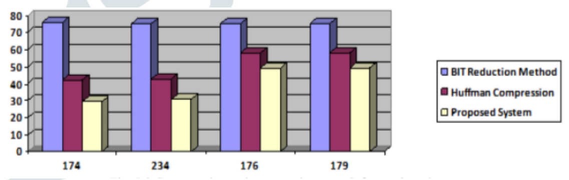

Fig 3: Comparison graph of random data set. [2]

COMPRESSION ALGORITHM

The following are the steps for compression:

1. Input the text data to be compressed.
2. Apply Dynamic Bit Reduction method to compress the data.
3. Find the unique symbol to compress the data further.
4. Create the binary tree with nodes representing the unique symbols
5. Apply Huffman coding to Finally compress the data.
6. Display the final result obtained in step V.

DECOMPRESSION ALGORITHM

The following are the steps for decompression:

1. Enter the compressed data as input.
2. Apply Inverse Huffman code to decompress the data.
3. Construct the binary tree representing nodes as data.
4. Replace the unique symbols with substitutes.
5. Apply reverse Dynamic Bit Reduction method to finally decompress the data.
6. Display the final result to the user.

2. **Shannon-Fano Coding (Modified):**

Different types of dynamic Shannon encoding show us the difference between dynamic and static encoding and teaches us about lower and upper bound. It also teaches us how dynamic coding is advantageous for variable length coding.

Method used by simple dynamic coding:
I=1,2……m

1. Running count of number of occurrences
2. Assignment of codewords
3. If si occurs in s1-si-1 encode si as ci
4. If not, then encode si as ci+ binary rep of si's index

Dynamic algorithm obtained by modifying Shannon coding uses at most (H + 1) m + O(n log m) bits and O(mnlogn) time to encode S.

Dynamic Shannon can be used in 3 spheres:
1-Dynamic Length restricted coding
2-Dynamic Alphabet coding
3-Dynamic coding with unequal costs

C. Image Compression using modified technique

In most images it is found that the neighboring pixels are correlated and hence have redundant data, thus it becomes vital for us to find less correlated representation to obtain a compressed image, this can be achieved by removing:

1. duplication from the signal source (image/video).
2. parts of the signal that will not be noticed by the signal receiver, namely the Human Visual System.

We have selected Huffman coding as our technique to obtain this kind of a decompressed image.

Huffman coding :

Image compression using Huffman is based on two observations:

1. More frequently occurred symbols will have shorter code words than symbol that occur less frequently.
2. The two symbols that occur least frequently will have the same length.

The following is the Huffman Coding and Decoding Algorithm:

1. Read the image on to the workspace of the mat lab.
2. Convert the given colour image into grey level image.
3. Call a function which will find the symbols (i.e. pixel value which is non-repeated).
4. Call a function which will calculate the probability of each symbol.
5. Probability of symbols are arranged in decreasing order and lower probabilities are merged and this step is continued until only two probabilities are left and codes are assigned according to rule that: the highest probable symbol will have a shorter length code.
6. Further Huffman encoding is performed i.e. mapping of the code words to the corresponding symbols will result in a compressed data.
7. The original image is reconstructed i.e. decompression is done by using Huffman decoding.
8. Generate a tree equivalent to the encoding tree.
9. Read input character wise and left to the table II until last element is reached in the table II.
10. Output the character encode in the leaf and return to the root, and continue the step9 until all the codes of corresponding symbols are known.

Fig 5: Input image

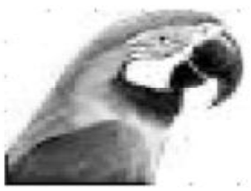
Fig 6: Decompressed image

III. LOSSY COMPRESSION

Lossless compression is where the data on reconstruction is found to be similar but not the same to the original data, the loss in accuracy is traded off for better efficiency. It is mainly used to compress multimedia applications such as image, audio and video which are used for activity of internet and media. One example of this is the JPEG image format.

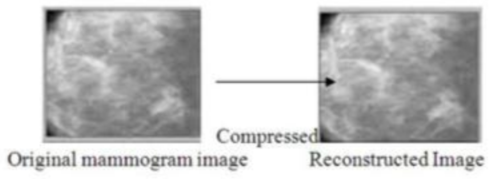

Fig 7: An example of lossy compression. [4]

A. A Lossy Image Compression using SPIHT tecnhique

The lossy compression technique that we will be using is the Set partitioning in hierarchical trees (SPIHT) technique. SPIHT is an image compression algorithm that exploits the inherent similarities across the sub-bands in a wavelet decomposition of an image, it is the best technique for progressive image transmission,

The reason for choosing this technique is because the algorithm provides the following:
1. Highest Image Quality
2. Progressive Image transmission
3. Fully embedded coded file
4. Simple quantization algorithm
5. Fast Coding/decoding
6. Exact bit rate coding
7. Error Protection

First the image is decomposed into four sub-bands. The decomposition process is repeated until we reach the final scale. Each decomposition consists of one low-frequency sub-band with three high-frequency sub-bands. [5]

SPIHT is the extension and efficient implementation of EZW algorithm.[5]

The entire algorithm follows three steps:

1. Sorting: The algorithm encodes the image in three lists such as LIP, LIS and LSP, where LIP contains the individual coefficients having the magnitudes smaller then the threshold value, LIS contains the overall wavelength defined in tree structure with magnitude less then threshold value, and LSP has the ones with magnitude higher then threshold, In sorting LIP list are verified and checked if they are important and all three list are tested using the equation:
   $$n_{max} = [log_2(max_{i,j}\{|C_{i,j}|\})]$$
   Only one coefficient is found important and is eliminated from subsets, then inserted into LSP or it will be inserted into LIP.
2. Refinement: Here the nth MSB of the coefficient in the LSP is taken as the final output, the value of n is decreased again sorting with refinement is applied until n=0.
3. Quantization

Once the encoding process is over, then the decoding process is applied. [5]

IV. PERFOMANCE METRICS

The following are the parameters used for the comparing the algorithms:

Entropy(H) is the measure of uncertainty in a Random variable, where Random variable is any variable or condition randomly chosen from a Sample space. It is measured in bits (logarithmic binary, powers of 2), nats (logarithmic unit, based on natural logarithms, powers of e) or bats(based on logarithms 10, powers of 10).

Average length is the total bit length calculated for sending any message after encoding.

Redundancy is the repeating bits.

$$\boxed{\text{ENTROPY} = \sum (P_K \cdot (\log_2 P_K))} \quad \boxed{\text{AVERAGE LENGTH} = \sum (P_k \times L_k)}$$

(i) (ii)

$$\boxed{\text{REDUNDANCY} = \left(\frac{\text{Avg Length-Entropy}}{\text{Entropy}}\right) \times 100}$$

(iii)

Minimax (sometimes MinMax or MM) is a decision rule used in decision theory, game theory, statistics and philosophy for *mini*mizing the possible loss for a worst case (*max*imum loss) scenario.

Greyscale Image is a pixel that carries only a single value, the intensity information of the pixel.

Compression ratio is defined as the ratio of size of the compressed file to the size of the source file.

Saving Percentage calculates the shrinkage of the source file as a percentage.

Compression ratio: C2/C1 *100% (iv)
Saving percentage: (C1-C2/C1) *100% (v)
C1= Size before compression C2= Size after compression

Prefix code is a type of code system (typically a variable-length code) distinguished by its possession of the "prefix property", which requires that there is no whole code word in the system that is a prefix (initial segment) of any other code word in the system.

PSNR is the ratio between maximum power of a signal and the noise corrupted signal that affects the reliability of the signal representation. It measures the quality of the image.

$$PSNR = 10.\log_{10}\left(\frac{255^2}{MSE}\right) \qquad \text{(vi)}$$

MSE is the metric used to verify the mean square error of the image, it is used to estimate the difference between two image in terms squared error value.

$$MSE = \frac{sum(sum(squaredErrorImage))}{(row \times columns)} \qquad \text{(vii)}$$

V. COMPARISON BETWEEN MODIFIED LOSSLESS AND LOSSY IMAGE COMPRESSION

Sr.No	LOSSLESS	LOSSY
1]	No loss in data	Varying amounts of data loss
2]	Higher Bandwidth	Lower Bandwidth
3]	Lower rate of compression	Higher rate of compression
4]	Uses techniques like dynamic bit reduction	Uses Progressive Image Transmission
5]	Exact File reproduction	Removes redundant data without affecting vital information
6]	Larger File Size	Smaller File Size
7]	Does not delete original bit from data	Reduces file size by removing bit permanently from data set.

Table 1. Comparison lossless vs lossy

Sr.No	Application	Type of Compression
1	WhatsApp	Lossy
2	ShareIt	Lossless
3	Instagram	Lossy
4	Xender	Lossless
5	YouTube	Lossy
6	OTA	Lossless
7	Facebook	Lossy

Table 2. Applications

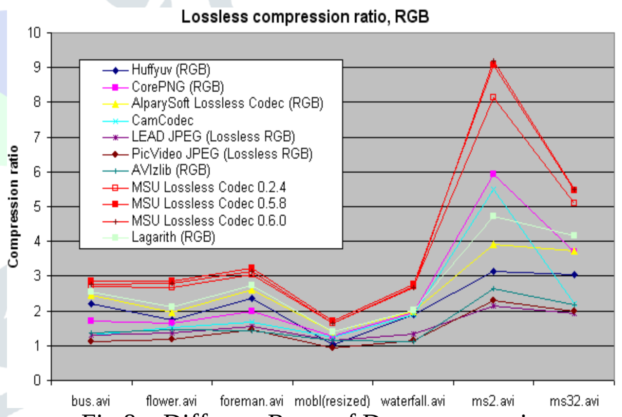

Fig 8x. Different Rates of Data compression

CONCLUSION

It was observed that day to day applications use different methods to store their data as referenced in Table 2.

For applications that have daily use and transmit conventional data like images and videos quotidian things use a lossy data compression. This is because since every bit of data is not vital. Pieces of information can be removed without losing integrity of the data.

On the other hand, there are places where every single bit of data is indispensable. In case of files like an apk file or an over the air update not a single bit of information can be lost. Changing a 1 to a 0 or vice-versa can change the complete functionality or result in bugs in the software.

Furthermore, different file formats have different ratios of compression as shown in figure

Thus at this point lossy compression is a more viable solution to data reduction but better metrics must be employed to reduce disruption of data.

REFERENCES

[1] Nidhi Dhawale, "Implementation of Huffman algorithm and study for optimization", *International Conference on Advances in Communication and Computing Technologies*, pp. 14-19, 2014

[2] Manjeet Kaur and Er. Upasna Garg, "Lossless Text Data Compression Algorithm Using Modified Huffman Algorithm", *International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 7*, pp. 1273-1276, July 2015

[3] Jagdish H. Pujar and Lohit M. Kadlaskar, "A new lossless method of Image Compression and Decompression using Huffman Coding Techniques", *Journal of Theoretical and Applied Information Technology*, pp. 18-23, 2010

[4] M.Hemalatha and S.Nithya, "A Through Survey on Lossy Image Compression Techniques", *International Journal of Applied Engineering Research ISSN Volume-5*, pp. 3326-3329 , 2016

[5] S.NirmalRaj, "SPIHT: A Set Partitioning in Hierarchical Tress Algorithm for Image Compression ", *EEE Department Sathyabama University*, pp. 263-270, 2015

[6] Student Victoria Cupet, Student Dan Cretu, Student Roland Dragoi, Student Bogdan Diaconu,"The Shannon–Fano Algorithm"*Faculty of Economic Cybernetics, Statistics and Informatics, Academy of Economic Studies*, pp 1-7

[7] Travis Gagie, "Dynamic Shannon Coding", *Student Member, IEEE*, pp 1-6