ELE725 Lab 2: Lossless Coding

Ashif Fahim
Department of Electrical, Computer and Biomedical Engineering
Ryerson University
Toronto, ON, Canada

Santos Bethany
Department of Electrical, Computer and Biomedical Engineering
Ryerson University
Toronto, ON, Canada

***Abstract*— In this lab the entropy of a source was calculated and a huffman encoder algorithm was coded to create prefix codes for the symbols in the source. The results showed that the decoded prefix code was able to output the same symbol as the one originally encoded demonstrating a lossless compression algorithm. The self information of each unique symbol of a source was calculated to determine the entropy-these probabilities are important information for creating histograms and optimizing compressions.**

I. Introduction/Theory *(Heading 1)*

Media compression is necessary practise in today's technological age where the size of the media source is reduced to save memory space but is still able to convey the same amount of information. Lossless compression specifically is when the data retrieved is exactly the same as the input (no data lost/removed). There are many algorithm that practise lossless compression, but the one that this lab focuses on is called the Huffman Coding. The Huffman algorithm creates unique prefix codes for each symbol and maps it to a lookup table[1]. This look up table will be used to decode and encode the data.

Entropy is the lower bounded number of bits per symbol need to encode a media source losslessly. This number is most often a theoretical value; in practise the number of bits per symbol required is higher compared to the calculated theoretical value.

II. Theory

A. Self - Information, Entropy, and Compression Ratio

The intuition behind self information is the association of less information for an event of higher probability and more information for an event of lower probability. With this idea in mind, we are able to take advantage of statistical redundancy in media source files when compressing. Self information of an event A is defined as[1]:

$$-log_2 P(A) = log_2 P(A)^{-1} \qquad (1)$$

where *P(A)* is the probability of event A.

The number of bits needed to encode a media source is lower bounded by its entropy. Entropy *H(s)* is simply the summation of each unique symbol's self information defined as follows [1]:

$$H(s) = P(A) log_2 P(A)^{-1} = -P(A) log_2 P(A) \quad (2)$$

Compression ratio is used to quantify the reduction of data of a media source file produced by a compression algorithm. The compression ratio is defined as follows[1]:

$$Compression\ Ratio = \frac{uncompressed\ data\ rate\ /\ size}{compressed\ data\ rate\ /\ size} \qquad (3)$$

B. Huffman Coding

Huffman coding is a lossless compression algorithm which is used to assign the most occurring symbols the least bits. Huffman coding uses Prefix codes which is a form of variable-length coding that assures the code assigned to one symbol will not be a prefix to code of another symbol. For example, let there be 'a', 'b', 'c', and 'd' symbols which have the corresponding codes 00, 01, 0, and 1. This brings ambiguity as the code of c (0) is the prefix of a (00) and b (01). If the encoded bit stream is '0001', the output could be '$cccd$', 'acd', 'ccb' or 'ab'. Prefix codes will not let this scenario happen.

Huffman coding will involve building a Huffman tree sorting the symbols by lowest frequency, and then assigning a Prefix code to each of the leafs of the tree. The steps to building a Huffman Tree is as follows:

1. Sort the symbols by the frequency going from lowest to highest.
2. Locate the two symbols with the lowest frequencies and join them to create a new parent node. This node will have a frequency that is equal to the frequencies of the child nodes.
3. Repeat step 2 until only the root node is left.
4. Label left branches with 0 and right branches with 1.
5. Follow the path from the root node to each leaf, and build the associated code to each symbol along the path.

After the Prefix codes are assigned, the sequence is looked at one character at a time and the associated Prefix code to the character is assigned to the output. For an image, the algorithm creates a new matrix with the same dimensions as the original image. Then each pixel is replaced by its associated Prefix code in the new encoded matrix with the same pixel location.

III. Methodology

A. Calculating the Entropy of a Source

MATLAB was used to perform all implementation and simulation of the experiments of this lab.

The first part of this lab is calculating the entropy of a given source. Below is the breakdown of the steps taken to do so:

1. Use *myEntropy(sequence)* function
   1.1. Find unique symbols if the input sequence.

    1.2. Calculate the probability of each symbol occurrence in the sequence.
2. Use *calcEntropy(probabilities)* function
    2.1. Implements equation *(2)* to calculate entropy using the probabilities from the output of the *myEntropy()* function.

The entropies of three sequences: (1) a string, (2) a grayscale image, (3) the Cb and Cr Chroma channels of a coloured image were calculated.

B. Huffman Coding Algorithm

In MATLAB, the Huffman Tree is implemented using a 2D matrix, where each row represents a symbol or combined node. In each row, the first column represents the probability of the symbol, the second column represents the parent node, third column represents the left child, fourth column represents the right child, the fifth column represents the Prefix code, and the sixth column represents the symbol as a string. If either of the child nodes or the parent nodes does not exist, a value of -1 will be put in its place.

Initially, the 2D Huffman Tree matrix will consist of the first four columns mentioned above. The Huffman Tree is built by first inserting the probability of each unique symbols of the sequence or image into a temporary array. When the two lowest probabilities are combined to create a parent node, the temporary array will remove the two child probabilities and add the parent probability. Afterwards, the parent node is added to the codebook including the index of the child nodes and letting its own parent node index to be -1. Simultaneously, both the child nodes' parent index will be changed from -1 to the index of the newly created parent index by searching through the 2D matrix with the same probabilities of the child nodes. If there is more than one symbol with the same probability, the algorithm will take the first symbol whose parent index is not -1, which means it has not been assigned a parent node yet. This is done so that each symbol with the same probability is treated differently by the algorithm. This process also ensures that a maximum of two child nodes have the same parent node and that each parent node has a unique left and right child that is not shared with any other parent node which follows the rules of a standard binary tree.

The next step will be to assign each node a Prefix code. This will involve including a fifth column initialized with a 0 for every symbol. Starting at the root node index, the left and right child will be assigned a 0 and 1, respectively. Then decrementing until every index has been reached, the current index will add its Prefix code of 0 or 1 depending on what type of child it is, with the Prefix code of its parent node. After this is done, a new matrix will be created from the old matrix by only the first *X* columns where *X* is the size of unique symbols. The algorithm will then take the unique symbols of the input and sort them by lowest frequency and assign each symbol to the sixth column of the matrix. Since this matrix is already sorted by the frequency of each symbol, the probabilities will be correctly assigned its correct symbol. This completes the codebook which has six columns and only unique symbols.

During the encoding and decoding process, the codebook will be used as look-up table for the symbols. To encode an image, a new matrix with the same dimensions will be created. Each pixel will be looked at, and replaced with the appropriate Prefix code in the new matrix which match the same coordinates. For a sequence, the Prefix codes will be appended to a single string.

For decoding an image, the process is similar as another matrix with the same dimensions will be created. Then each Prefix code will be compared with in the codebook to find the appropriate symbol which will be placed in the new matrix with the same coordinates. Decoding a sequence/string differs slightly, as it will look at the first character and see if such a Prefix code exists in the codebook. If it does not, then the next character is added to the current character and checked again. This is repeated until a Prefix code is found, which will then be placed in the output sequence/string.

IV. Results

TABLE I. Entropy and Compression Ratio Results

Source	*Entropy*	*Compression Ratio*
String	3.99	-
Gray scale image	7.45	1.07
Cb channel of a coloured image	5.03	1.51
Cr channel of a coloured image	5.25	1.52

a. Assuming uncompressed bits per pixel is 8bpp for grayscale, Cr, and Cb channel

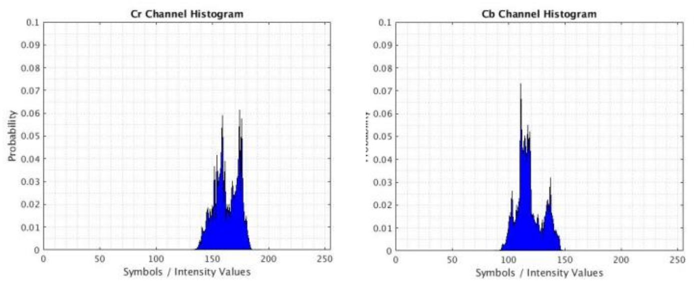
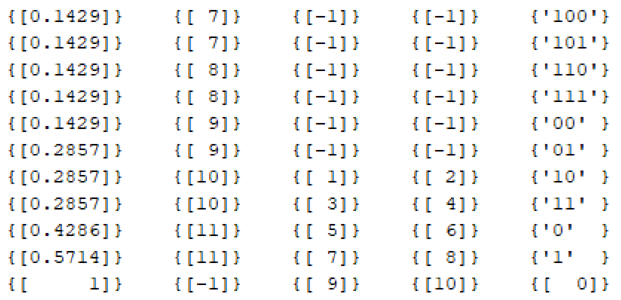
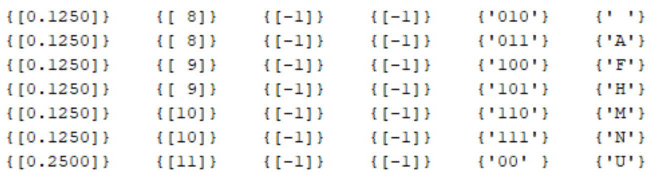


Figure 1. (a) Cr and (b) Cb Channel Histogram



```
{[0.1429]}    {[ 7]}    {[-1]}    {[-1]}    {'100'}
{[0.1429]}    {[ 7]}    {[-1]}    {[-1]}    {'101'}
{[0.1429]}    {[ 8]}    {[-1]}    {[-1]}    {'110'}
{[0.1429]}    {[ 8]}    {[-1]}    {[-1]}    {'111'}
{[0.1429]}    {[ 9]}    {[-1]}    {[-1]}    {'00' }
{[0.2857]}    {[ 9]}    {[-1]}    {[-1]}    {'01' }
{[0.2857]}    {[10]}    {[ 1]}    {[ 2]}    {'10' }
{[0.2857]}    {[10]}    {[ 3]}    {[ 4]}    {'11' }
{[0.4286]}    {[11]}    {[ 5]}    {[ 6]}    {'0'  }
{[0.5714]}    {[11]}    {[ 7]}    {[ 8]}    {'1'  }
{[     1]}    {[-1]}    {[ 9]}    {[10]}    {[  0]}
```

Figure 2. 2D Huffman Tree Matrix for a Sequence



```
{[0.1250]}    {[ 8]}    {[-1]}    {[-1]}    {'010'}    {' '}
{[0.1250]}    {[ 8]}    {[-1]}    {[-1]}    {'011'}    {'A'}
{[0.1250]}    {[ 9]}    {[-1]}    {[-1]}    {'100'}    {'F'}
{[0.1250]}    {[ 9]}    {[-1]}    {[-1]}    {'101'}    {'H'}
{[0.1250]}    {[10]}    {[-1]}    {[-1]}    {'110'}    {'M'}
{[0.1250]}    {[10]}    {[-1]}    {[-1]}    {'111'}    {'N'}
{[0.2500]}    {[11]}    {[-1]}    {[-1]}    {'00' }    {'U'}
```

Figure 3. Codebook for a Sequence

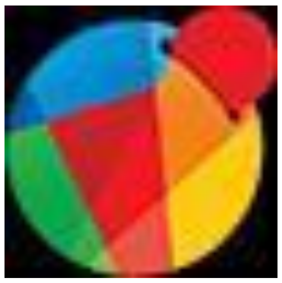


Figure 4(a) Original Image

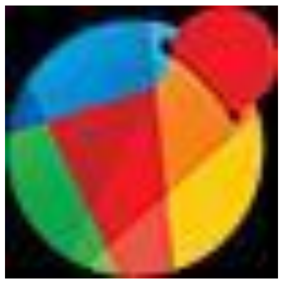


Figure 4(b) Huffman Coded Image

TABLE II. Comparing Entropy and Average Bit Size of Channels

Channel	*Entropy*	*Compression Ratio*	*Average Number of Bits*	*Compression Ratio*
Cb	5.3012	1.509	5.8912	1.357
Cr	5.2512	1.523	5.9615	1.341

V. Discussion

A. Entropy

Results from Table 1 shows that the lowest possible amount of bits to represent a pixel for a gray scale image, and chroma channels of a coloured image. Since it is assumed that the uncompressed bit size of the gray scale image is 8 bpp and that the compressed bit size is the entropy of 7.45, it is intuitive for the compression ratio to near 1 representing a little to no change in bit size. Applying the same assumption to the chroma channels, their entropy is slightly lower suggesting a higher compression ratio.

The plots from Figure 1 was obtained by plotting probability of each unique intensity values of the chroma channels (output from the *myEntropy()* function) with the unique intensity values of the chroma channels. This then produces what is called a histogram. A histogram shows the distribution of the frequencies ("how often") a symbol appears in the data. In a histogram, we are able to see which colour values are used the most, the least, and not at all in the media source. This information is useful when determining the best way to optimized compression approach of the source. For example, in Figure 1(b) the Cb Channel can be encoded in such a way where the least information will be associated with the intensity value 111 because it has the highest probability, and the most information will be associated with intensity values with lower probabilities (0.001-0.03).

B. Huffman Coding

For a sequence such as 'HUFF MAN' (the gap is intended to show algorithm accounts for spaces), the encoded single will be '10100100100010110011111' where the 2D Huffman Tree matrix for this particular sequence can be seen in Figure 2. The Huffman Tree matrix includes only 5 columns excluding the actual symbol for each probability. This is done because the matrix also includes probabilities of parents nodes which do not have a symbol associated with them. It should also be noted that rows which have -1 for both left and right child indexes, are leaf nodes and are unique symbols. The root index will be assigned a parent index of -1 as it does not have a parent node, and its Prefix code is 0 as integer whereas all other nodes have a Prefix as a string because the branches themselves are assigned a 0 or 1 to create the Prefix codes and not the nodes.

Figure 4 shows the codebook which contains only unique symbols and assigns symbols accordingly with its proper probability.

The Huffman Tree and codebook for images are similar to those for sequences, with the different being instead of character symbols, they will be integers represented as strings.

Figure 4(a) shows an image of resolution 50x50, which allows the algorithm to be computed quickly, and Figure 4(b) which shows the same image whose Cb and Cr channels encoded and decoded using the Huffman encoder. The images show almost no notable differences. Table 2 shows the entropy of the image and the average number of bits of the encoded image. The average number of bits was calculated by taking the size of bits in each pixel location of the encoded image and dividing the total number of pixels where the entropy was calculated using the function from part 1 for the image shown in Figure 4(a). The average number of bits was slightly higher than the entropies leading to smaller compression ratios.

VI. Conclusion

In conclusion, it was found the compression ratio of the tested grayscale image was close to 1 which means compressing this image would have little to no effect in reducing the file size. For colour image, the Cb and Cr channels are encoded; their compression ratios were found to be around 1.5. This is slightly better, but still very low, higher compression ratios are desired. When applying the Huffman encoder to test images, the average number of bits were found to be slightly higher than the entropies which led to smaller compression ratios. This experiment explored lossless compression ratios which meant no data was lost but the compression ratios would be relatively low. Higher compression ratios can be obtained using lossy compression algorithms which can studied next as the basic concepts such as entropy, compression ratio, Prefix codes and Huffman encoding have been established in this experiment. However, the application of Huffman encoding was successful as a test sequence and image were both encoded and decoded to give the same result without any data loss which was the objective of the experiment.

References

[1] Z. Li and M. Andrew, *Fundamentals of Multimedia*, 1st ed. New Jersey: Pearson Education, 2004.