Indian Institute of Technology, Ropar

CS517: Digital Image Processing and Analysis
Karanraj Mehta
2021CSB1100

LZW Compression

April 20, 2023

Overview

This LZW Compression Project aims to implement the Lempel-Ziv-Welch (LZW) algorithm to compress grayscale images. The LZW algorithm is a lossless compression algorithm that achieves high compression ratios by replacing frequently occurring substrings with shorter codewords.

The project includes an encoder that compresses the input image into an encoded image and a decoder that takes the encoded image and reconstructs the original image. Several utility functions have been created to calculate the compression ratio, maximum achievable compression and entropy for the image.

Goals

The goals of this project are to:

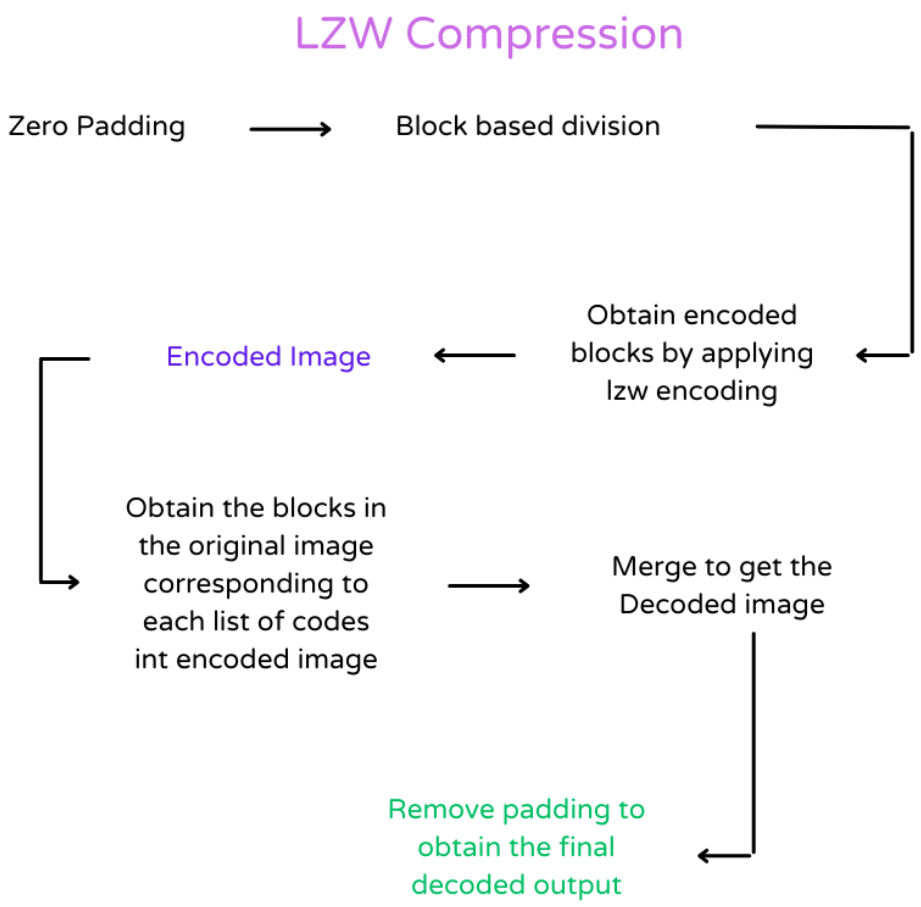
1. **Implement the LZW algorithm for grayscale image compression**
2. **Obtain the compressed image with the quality same as the original**
3. **Create lzw encoder and decoder for the compression algorithm**
4. **Calculate the compression ratio, entropy and maximum compression achievable**
5. **Analyze the algorithm for different images**

Specifications

The program takes a grayscale image, block size and the maximum allowed dictionary size from the user. The lwz encoder uses these and calculates the encoded images, a list of lists where each list corresponds to the lzw encoding of one block in the image. This encoded image is then written in a text file.

The lzw decoder takes this encoded image and decodes it to get the decoded image. Since lzw is, a lossless compression, the obtained RMSE between the original and the decoded image is 0.

We use the other utility functions to calculate the values for compression ratio, entropy and maximum achievable compression.



The encoder first pads the image at the last row/column to make it divisible by the block size. Additionally, if the block size is -1, the whole image is treated as a single block. Next, the image is divided into blocks by slicing the 2D numpy array. Then we apply the lzw compression algorithm on each of these blocks after flattening them. We maintain a few variables such as currently_recognized to store the currently recognized pattern, encoded_ouput to store the encoded output for the currently recognized pattern and code_dict, which stores the generated codes in a dictionary.

In each iteration, we check if the currently recognized pattern is present in the code dictionary. If yes, we update the encoded_output, otherwise, we add it to our dictionary.

For the decoder, we use the lzw decoding algorithm. Again, we maintain a dictionary for storing the codes. For each list in the encoded image, we calculate the corresponding block. Taking all these blocks together we obtain the decoded image.

This was just a high level discussion of the program. There are many finer details that I have explained in detail within the code through comments.

Analysis and Result

LZW compression primarily utilizes two types of redundancies: spatial redundancy and statistical redundancy.

Spatial redundancy refers to the fact that in an image, adjacent pixels tend to have similar values. LZW compression takes advantage of this redundancy by encoding repeating patterns of pixels as a single code.

Statistical redundancy refers to the fact that in any given data, certain symbols or combinations of symbols are more common than others. LZW compression utilizes statistical redundancy by assigning shorter codes to more frequently occurring patterns, thereby reducing the overall number of bits required to represent the data.

The program was test with the images provided in the data set. For the images Images with high spatial and statistical redundancy showed large compression ratios.

The algorithm is efficient, taking less than a minute to compress and decompress large images. There was no loss of quality of the compressed images. The RMSE in each case was found to be zero as expected.

Outputs for a few test cases:

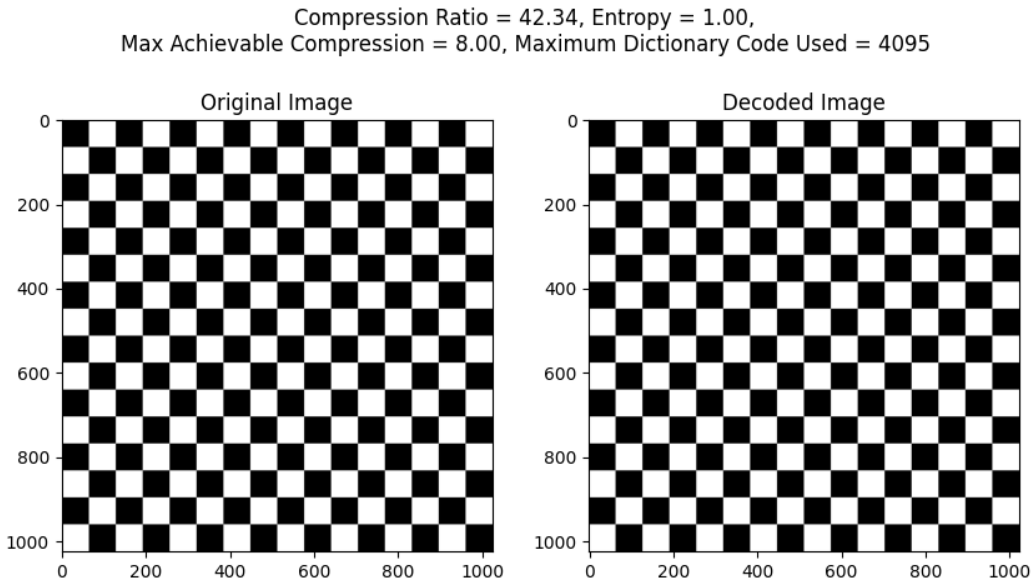
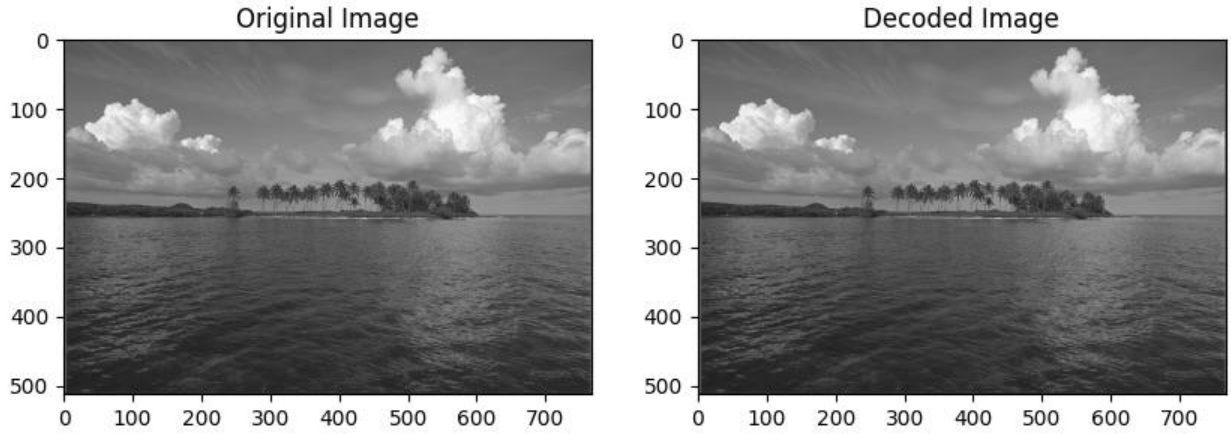


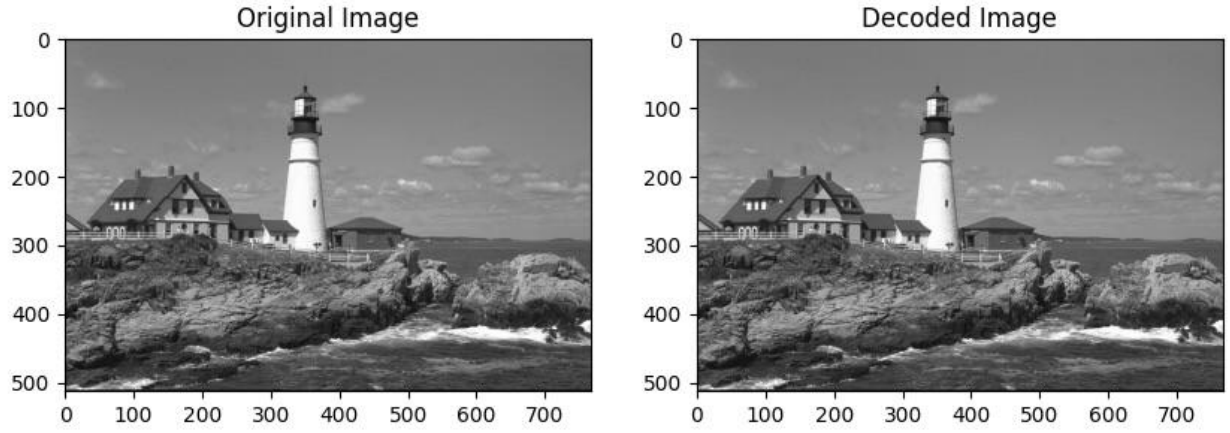
Figure 1: Image with high redundancy showing a very large compression ratio

These are a few more output files. From these outputs we can observe that lzw compression is only useful for images with high spatial and statistical redundancies.

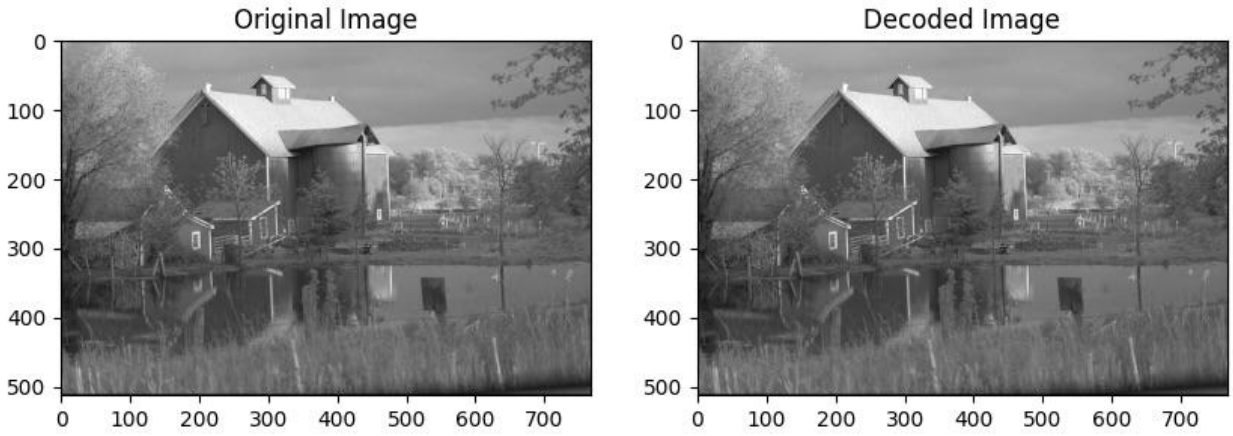
Compression Ratio = 1.07, Entropy = 6.63,
Max Achievable Compression = 1.21, Maximum Dictionary Code Used = 4095



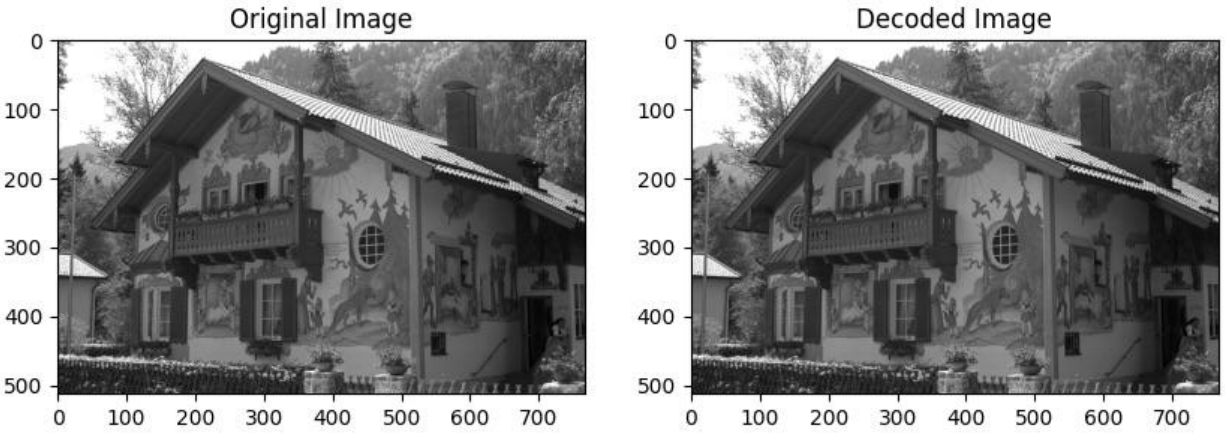
Compression Ratio = 1.02, Entropy = 6.62,
Max Achievable Compression = 1.21, Maximum Dictionary Code Used = 4095



Compression Ratio = 1.09, Entropy = 6.74,
Max Achievable Compression = 1.19, Maximum Dictionary Code Used = 4095



Compression Ratio = 1.10, Entropy = 6.54,
Max Achievable Compression = 1.22, Maximum Dictionary Code Used = 4095



Usage

- Use python version >= 3.10
- Install the required libraries: Numpy, OpenCV and Matplotlib
- Run the program using python A4_LZW_2021CSB1100_Karanraj.py