Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
MoE, UGC & AICTE Approved
NAAC A++ Accredited

Department of Electronics and Communication Engineering

III IA EVALUATION REPORT
for
BLENDED LEARNING PROJECT BASED LEARNING

A report submitted by

Name of the Students	**ALDRIN G, BRIJIN G**
Register Numbers	**URK22EC1019, URK22EC1021**
Subject Name	**DIGITAL COMMUNICATION**
Subject Code	**22EC2016**
Date of Report submission	**28/10/24**

Project Rubrics for Evaluation

First Review: to be awarded for **10 Marks** – PPT should have four slides (Title page, Introduction, Circuit/Block Diagram, and Description of Project).

Second Review: to be awarded for **10 Marks** – PPT should have three slides (Description of Circuit/Concept, Design of Circuit in Tools/Design of Flow graph, and Partial Results)

Third Review: to be awarded for **10 Marks** – PPT should have two slides (Tool based simulation/Hardware based simulation/Concept based interaction/ Case Study)

Fourth Review: to be awarded for **10 Marks** –PPT should have two slides (Output Results & Discussion)

[NOTE: The instructions are subjected to change if required to combine either of two reviews and to be evaluated for 20 Marks.]

Total marks: _____/ 40 Marks

Signature of Faculty with date:

IMPLEMENTATION OF SHANNON FANO CODEC ON LAB IMAGES

a project report submitted by

ALDRIN G(URK22EC1019)

BRIJIN G(URK22EC1021)

**in partial fulfillment for the award of the degree
of**

BACHELOR OF TECHNOLOGY

For

DIGITAL COMMUNICATION

– PROJECT BASED LEARNING (22EC2016)

under the supervision of

Dr. S. Merlin Gilbert Raj

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Deemed-to-be-University)

Karunya Nagar, Coimbatore - 641 114. INDIA

October 2024

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **“IMPLEMENTATION OF SHANNON FANO CODEC ON LAB IMAGES”** is the bonafide work of **“ALDRIN G (URK22EC1019) and BRIJIN G (URK22EC1021)”** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
DR.JUDE HEMANTH	DR.S.MERLIN GILBERT RAJ
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of ECE	Department of ECE
School of E&T	School of E&T

Submitted for the III Internal-Project Based Learning Viva Voce held on _______________.

Internal Examiner

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In the digital age, efficient data compression techniques are essential for optimizing storage and transmission of information, particularly in fields like image processing. This project report delves into the implementation of Shannon-Fano coding, a foundational method for lossless data compression, applied specifically to LAB images.

Shannon-Fano coding, developed by Claude Shannon and Robert Fano in the year 1949, is based on the principle of entropy and variable-length coding, where more frequent symbols are represented with shorter codes, enhancing compression efficiency. The importance of effective image compression becomes especially pronounced in medical imaging, where large datasets are generated and the ability to transmit high-quality images swiftly can significantly impact diagnosis and treatment.

This report outlines the methodology used to implement Shannon-Fano coding for LAB images, highlighting the differences in image characteristics and the implications for compression performance. By analyzing the results, we aim to demonstrate the effectiveness of Shannon-Fano coding in reducing file sizes while maintaining image integrity. The findings of this project not only contribute to the field of image processing but also provide valuable insights for applications requiring efficient data management.

Through this exploration, we hope to underscore the significance of innovative compression techniques in enhancing the usability and accessibility of digital images in various domains.

CHAPTER 2

PROBLEM STATEMENT

"Laboratory imaging systems often produce large data files, leading to high storage and transmission costs. An effective compression method is required to reduce file sizes without compromising the quality needed for accurate analysis."

Shannon-Fano coding can provide an effective solution for compressing laboratory imaging data files, as it balances file size reduction with quality retention, which is crucial for accurate analysis.

MARKET SURVEY:

Target Sectors:

Hospitals and Medical Centers, Radiology Departments, Medical Device Manufacturers

Current Challenges:

Very large Data Volume, High Storage Costs, Image Quality Preservation

Current Compression Techniques:

- **JPEG2000 and Lossless JPEG:** Widely used, particularly in DICOM, but they have limitations in terms of computational efficiency and compression ratio.
- **Other Coding Techniques:** Often used but could be inefficient for certain types of data.
- **Proprietary Compression Algorithms:** Some organisations use their own compression techniques, making integration with other systems difficult.

OBJECTIVES:

1. Optimize Data Storage and Transmission Costs

- **Objective**: Significantly reduce the storage requirements of large imaging datasets by compressing file sizes using Shannon-Fano coding.
- **Goal**: Achieve a measurable decrease in file size (e.g., reducing file size by at least 50%) to lower storage and transmission costs while maintaining data quality.

2. Maintain Image Quality for Accurate Analysis

- **Objective**: Ensure that the compression method does not compromise the data integrity required for accurate analysis.
- **Goal**: Evaluate the quality of decompressed images by calculating PSNR and maintaining it above a set threshold (e.g., PSNR > 30 dB) to ensure diagnostic or analytical accuracy is retained.

3. Implement Efficient Coding and Decoding Process

- **Objective**: Develop an effective Shannon-Fano coding algorithm for encoding and decoding image data, optimized for computational efficiency and speed.
- **Goal**: Minimize processing time for both encoding and decoding by designing an efficient coding tree and implementing streamlined encoding and decoding functions.

4. Develop Probability-Based Code Allocation

- **Objective**: Create and assign variable-length binary codes based on the probability distribution of pixel values or feature intensities in the image.
- **Goal**: Achieve an optimal code length allocation where more frequent symbols are assigned shorter codes, contributing to a higher compression ratio.

5. Measure and Analyze Compression Efficiency

- **Objective**: Calculate key performance metrics such as compression ratio, entropy, average code word length, and efficiency.

- **Goal**: Generate a comprehensive analysis report that includes these metrics to quantify the effectiveness of the Shannon-Fano coding process for laboratory imaging data.

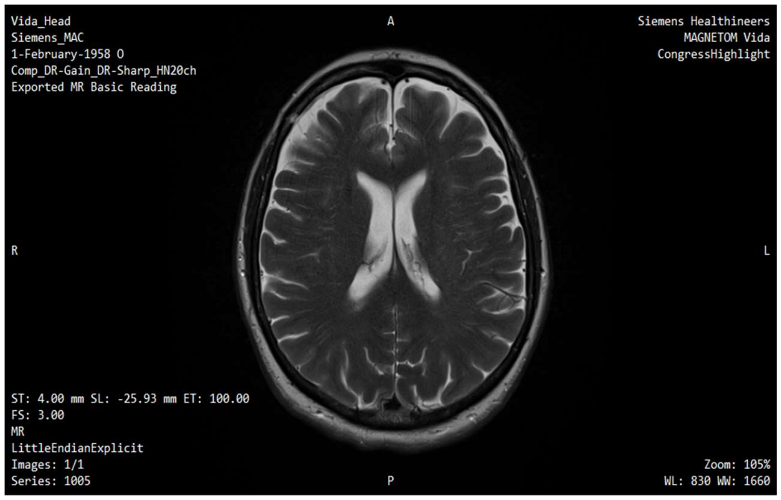
6. Ensure Scalability and Adaptability

- **Objective**: Design the compression system to handle varying image types, resolutions, and data sizes typically found in laboratory settings.
- **Goal**: Develop a scalable solution that can be adapted to compress and decompress different laboratory imaging data formats with minimal adjustments.

7. Evaluate and Compare Compression Techniques

- **Objective**: Compare Shannon-Fano coding results with other compression methods (e.g., Huffman coding, JPEG compression) to determine its suitability.
- **Goal**: Identify scenarios where Shannon-Fano coding provides superior performance, as well as any limitations, to make informed recommendations for its use in laboratory imaging.

LAB IMAGE :



INFERENCE:

LAB Images must retain quality as it carries lot of information and the information regarding the condition and abnormalities in the body of a patient.

CHAPTER 3

DESCRIPTION OF THE PROJECT

The objective of this project is to implement an efficient image compression system based on Shannon-Fano encoding, aimed at minimizing storage space and transmission bandwidth while preserving image quality.

Project Description

1. **Objective**:
   - To compress LAB images using Shannon-Fano encoding and analyze compression metrics such as entropy, average codeword length, efficiency, and Peak Signal-to-Noise Ratio (PSNR) between the original and decompressed images.
2. **Implementation of Shannon-Fano Encoding:**
   - **Load LAB Image:**
     Load the uncompressed LAB Image of DICOM Format.
   - **Frequency Analysis**:
     Calculate the frequency of each symbol in the data set (e.g., each pixel value in an image).
   - **Probability Calculation**:
     Calculate the probability of each symbol by dividing its frequency by the total number of symbols.
   - **Sorting**:
     Sort the symbols in descending order of probability.

- **Recursive Partitioning**:

  Split the list of symbols into two groups with approximately equal total probability. Assign a binary digit ('0' to one group and '1' to the other).

- **Output Compressed Data**:

  Store or transmit the encoded binary sequence along with a lookup table for decoding. This table maps each symbol to its Shannon-Fano code.

- **Calculate Compression Metrics:**
  - ❖ Entropy (H)
  - ❖ Average Code Word Length
  - ❖ Efficiency (η)

- **Visualize Data in Charts & Graphs:**

  Entropy vs Average Code Word Length Graph is plotted to find efficiency and Symbol Count vs Code Word Length Graph is plotted to find the relation between symbol count and code word length.

3. **Implementation**:

- The program uses libraries like PIL (Python Imaging Library) for image processing, NumPy for numerical operations, and OpenCV for PSNR calculations.

- The program creates a structured folder, Results1, where it saves compressed, decompressed images and analysis files such as symbol statistics, raw data, and visualizations.

4. **Outputs**:

    - **Compressed File**: Contains the binary representation of the original image.
    - **Decompressed Image**: Saved as a .jpg file, allowing a visual comparison with the original.
    - **Metrics and Analysis**: Includes PSNR value, entropy, average code length, and efficiency percentage.
    - **Plots**: Visual charts of symbol counts, code lengths, and entropy ratios, saved in the results folder for easy reference.

Tools and Libraries:

1) **Python:** The primary programming language for the project.
2) **NumPy:** For handling numerical operations and signal processing.
3) **Matplotlib:** For visualizing different metrics and compression results.
4) **CV Zone:** For image processing.
5) **Pydicom:** To read dicom images in python environment

CHAPTER 4

CONCEPT INVOLVED

1. DATA COMPRESSION

Data compression refers to the process of encoding information using fewer bits than the original representation. In the context of images, effective compression is essential for handling large files in both consumer applications (like photography) and critical fields (like medical imaging).

Compression techniques can be classified into two main categories:

- **Lossy Compression**: This method reduces file size by removing some data, resulting in a loss of quality. Commonly used in multimedia files (like JPEG for images), lossy compression is often acceptable for applications where slight degradation in quality is permissible.
- **Lossless Compression**: In contrast, lossless compression allows for the exact original data to be reconstructed from the compressed data. This is crucial in applications where data integrity is paramount, such as in medical imaging, where every detail is important for diagnosis.

2. SHANNON-FANO CODING

2.1 Overview

Shannon-Fano coding is a method for lossless data compression that was developed by Claude Shannon and Robert Fano.

2.2 How It Works

The process of Shannon-Fano coding can be summarized in several steps:

1. **Frequency Analysis**: First, the frequency of each symbol (in this case, pixel values) is determined. This frequency distribution is essential for the next steps.
2. **Tree Construction**: The symbols are sorted by frequency, and a binary tree

3. **Code Assignment**: Once the tree is built, binary codes are assigned to each symbol based on their position in the tree. More frequent symbols receive shorter codes, while less frequent symbols get longer codes.

4. **Encoding and Decoding**: The original data can then be encoded into a compressed format using the assigned binary codes. The decoding process reconstructs the original data using the same tree structure.

3. LAB IMAGES

3.1 Characteristics of LAB Images

LAB images, such as MRI, CT, and X-ray scans, often contain grayscale pixel values representing various tissue types. The intensity of each pixel can correspond to different anatomical structures or pathologies, making the precise representation of details critical for diagnosis.

3.2 Importance of Lossless Compression

For medical images, lossless compression is essential. Any loss of detail can result in misdiagnosis or failure to detect critical conditions. Therefore, compression techniques used in medical imaging must ensure that all relevant information is preserved.

3.3 Common Formats

Common formats for medical images include DICOM (Digital Imaging and Communications in Medicine), which standardizes the handling and transmission of medical images, ensuring compatibility across various imaging devices and software.

4. PERFORMANCE METRICS

4.1 Compression Ratio

The compression ratio is a critical metric in evaluating the effectiveness of a compression algorithm. Higher ratios indicate more effective compression, but it is important to balance compression with quality.

4.2 Processing Time

The time taken for both encoding and decoding processes is essential for assessing the algorithm's efficiency. Fast processing times are particularly important in real-time applications and large-scale image databases.

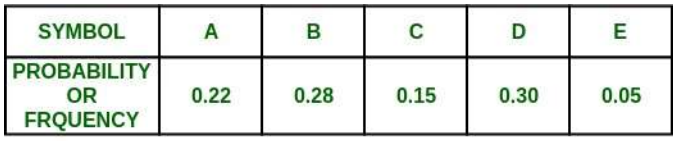
4.3 Image Quality Assessment

For DICOM images, metrics such as Peak Signal-to-Noise Ratio (PSNR) can be used to quantify quality. In medical images, expert evaluations may be necessary to ensure that critical diagnostic details remain intact after compression.

CHAPTER 5

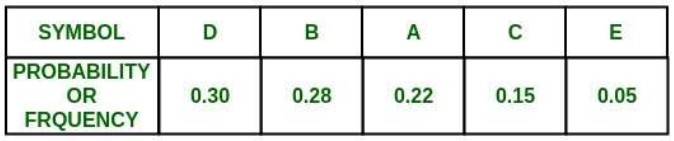
WORKING OF SHANNON FANO CODING

➢ SHANNON-FANO TREE



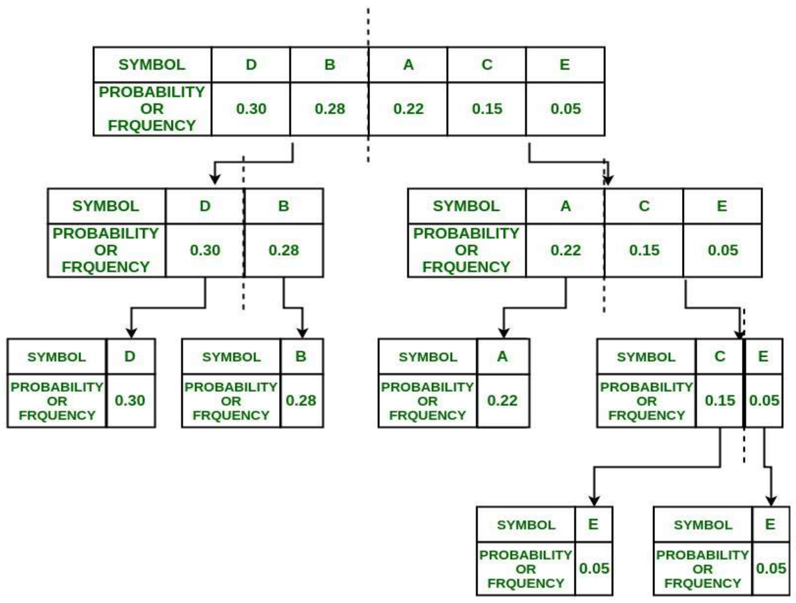
SYMBOL	A	B	C	D	E
PROBABILITY OR FRQUENCY	0.22	0.28	0.15	0.30	0.05

THE SYMBOLS AND THEIR PROBABILITY / FREQUENCY
ARE TAKEN AS INPUTS.
(In case of Frequency, the values can be any number)

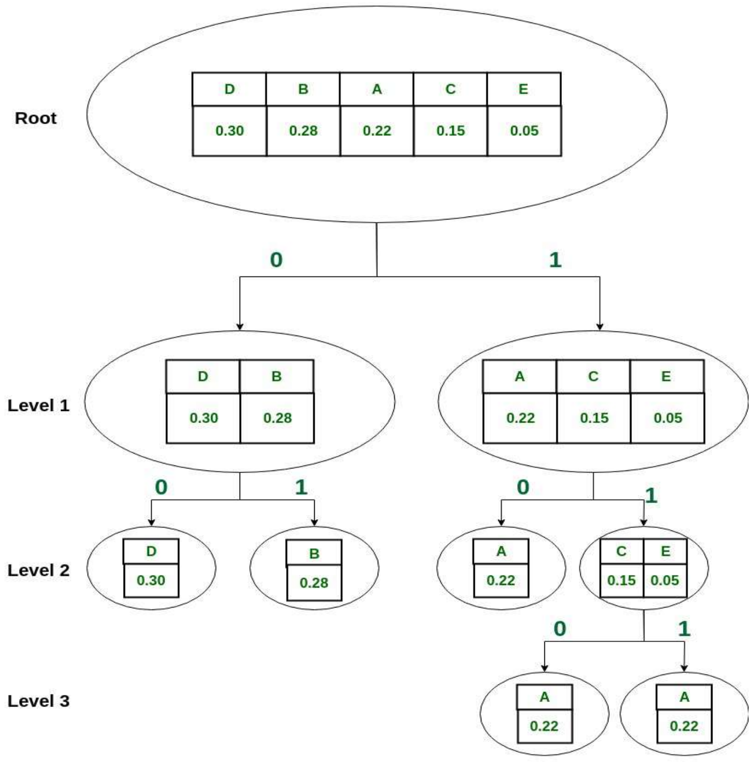


SYMBOL	D	B	A	C	E
PROBABILITY OR FRQUENCY	0.30	0.28	0.22	0.15	0.05

INPUTS ARE SORTED ACCORDING
TO THEIR PROBABILITY / FREQUENCY
(Here they are sorted according to their probability)

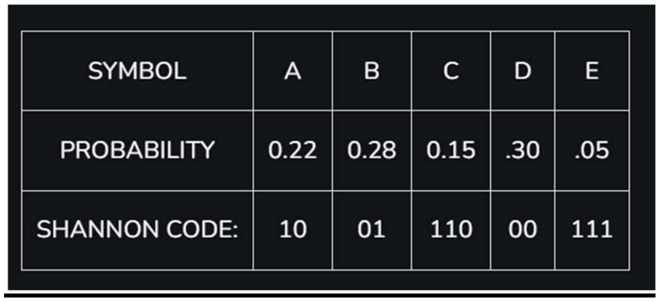

SYMBOL D B A C E
PROBABILITY OR FRQUENCY 0.30 0.28 0.22 0.15 0.05
SYMBOL D B
PROBABILITY OR FRQUENCY 0.30 0.28
SYMBOL A C E
PROBABILITY OR FRQUENCY 0.22 0.15 0.05
SYMBOL D
PROBABILITY OR FRQUENCY 0.30
SYMBOL B
PROBABILITY OR FRQUENCY 0.28
SYMBOL A
PROBABILITY OR FRQUENCY 0.22
SYMBOL C E
PROBABILITY OR FRQUENCY 0.15 0.05
SYMBOL E
PROBABILITY OR FRQUENCY 0.05
SYMBOL E
PROBABILITY OR FRQUENCY 0.05

THE SYMBOLS ARE CONTINUED TO BE DIVIDED INTO TWO
TILL EACH SYMBOL BECOME SEPARATED



TREE AFTER STEP 4

Shannon Fano Encoded Table:



SYMBOL	A	B	C	D	E
PROBABILITY	0.22	0.28	0.15	.30	.05
SHANNON CODE:	10	01	110	00	111

CHAPTER 6

FORMULAE USED

1. Avg. Code Word Length:

$$\text{Lavg} = \sum_{i=0}^{k} \text{Pi Li}$$

Where: Pi = Probability

Li = Length of the i^{th} code word

2. Entropy:

$$\text{H} = \sum_{i=1}^{k} Pi\ log_2\ \frac{1}{Pi}$$

3. Efficiency:

$$\eta = \frac{Entropy\ (H)}{Average\ code\ word\ length(Lavg)}$$

4. Peak Signal-Noise Ratio (PSNR):

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{\text{MSE}}\right)$$

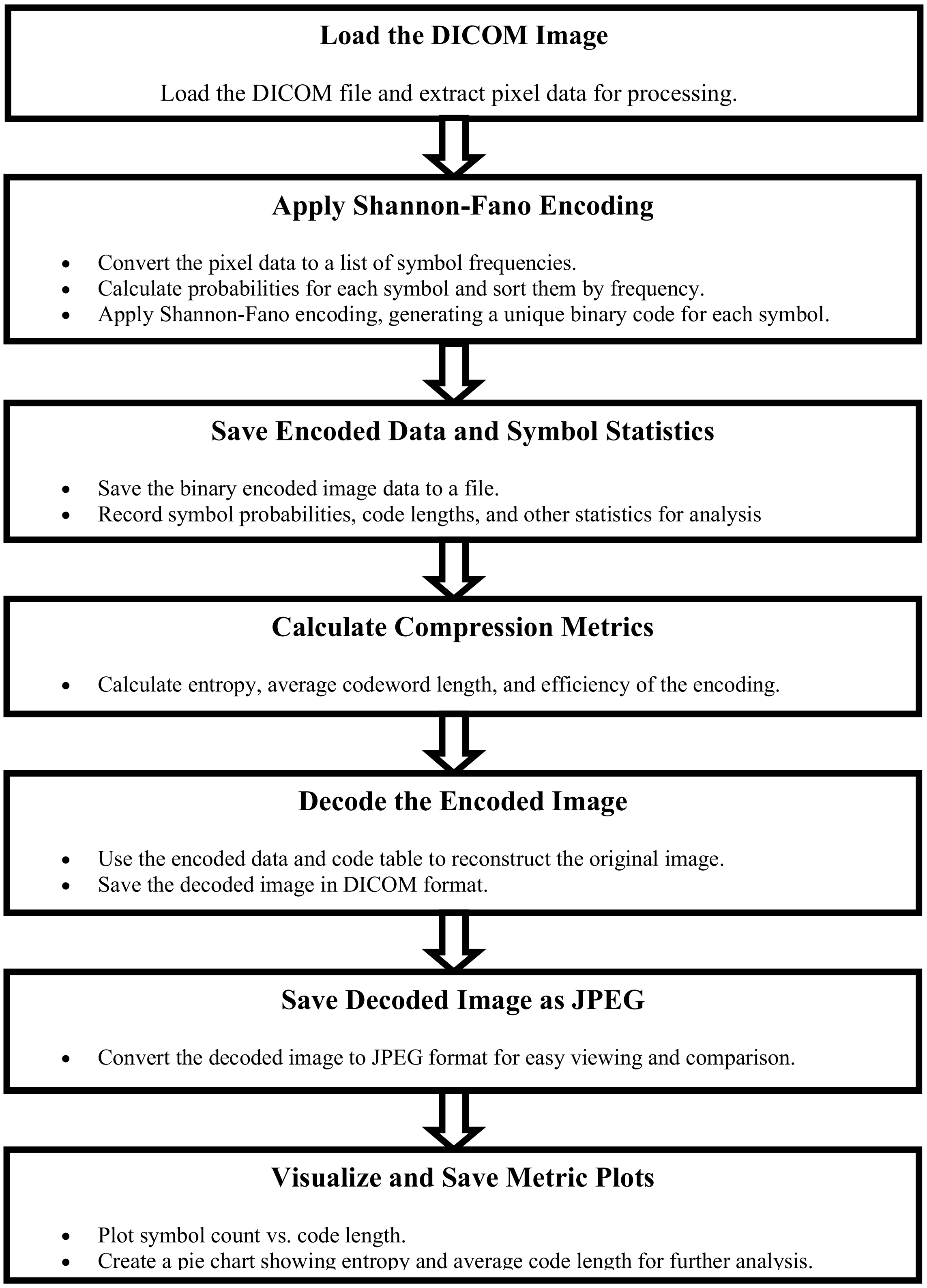
Where,

$$\text{MSE} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\big(I(i,j) - K(i,j)\big)^2$$

MAX = 255 (Maximum Pixel Value of Image)

CHAPTER 7

BLOCK DIAGRAM / FLOW CHART



Load the DICOM Image

Load the DICOM file and extract pixel data for processing.

Apply Shannon-Fano Encoding

- Convert the pixel data to a list of symbol frequencies.
- Calculate probabilities for each symbol and sort them by frequency.
- Apply Shannon-Fano encoding, generating a unique binary code for each symbol.

Save Encoded Data and Symbol Statistics

- Save the binary encoded image data to a file.
- Record symbol probabilities, code lengths, and other statistics for analysis

Calculate Compression Metrics

- Calculate entropy, average codeword length, and efficiency of the encoding.

Decode the Encoded Image

- Use the encoded data and code table to reconstruct the original image.
- Save the decoded image in DICOM format.

Save Decoded Image as JPEG

- Convert the decoded image to JPEG format for easy viewing and comparison.

Visualize and Save Metric Plots

- Plot symbol count vs. code length.
- Create a pie chart showing entropy and average code length for further analysis.

CHAPTER 8

PYTHON CODE

Python Code to Implement Shannon Fano Codec On LAB Images

```
import re
import numpy as np
from PIL import Image
import pydicom
import collections
import os
import math
import matplotlib.pyplot as plt
import cv2

name = ''

# Function to create the Results folder if not already present
def create_results_folder(n):
    folder_name = f'Results{n}'
    if not os.path.exists(folder_name):
        os.makedirs(folder_name)
    return folder_name

# Function to calculate PSNR
def calculate_psnr(image1_path, image2_path):
    img1 = cv2.imread(image1_path)
    img2 = cv2.imread(image2_path)

    if img1 is None:
        raise ValueError(f"Error: Could not load image at {image1_path}")
    if img2 is None:
        raise ValueError(f"Error: Could not load image at {image2_path}")

    if img1.shape != img2.shape:
        raise ValueError("Input images must have the same dimensions.")

    img1 = img1.astype(np.float32)
    img2 = img2.astype(np.float32)

    mse = np.mean((img1 - img2) ** 2)
    if mse == 0:
        return float('inf')

    max_pixel = 255.0
    psnr = 20 * np.log10(max_pixel / np.sqrt(mse))

    return psnr

print("\nShannon Image Compression Program")
print("============================================================
===")

num_times = int(input("\nEnter the number of times to run the process: "))

for i in range(1, num_times + 1):
    results_folder = create_results_folder(i)


print("\n===========================================================
=====")

    print(f"\nRunning process for image {i}")

    h = int(input("\nInput RGB(1)/DICOM(2): "))

    if h == 1:
        file = input("\nEnter the filename: ")
        name = file
        original_image = Image.open(file)
        original_image.save(os.path.join(results_folder, 'original.jpg'))  # Save the original
image
        my_string = np.asarray(original_image, np.uint8)
        sudhi = my_string
        shape = my_string.shape
        print("\n\nEntered image data is:")
        message = str(my_string.tolist())
    elif h == 2:
        file = input("\nEnter the DICOM filename: ")
        dicom_image = pydicom.dcmread(file)
        pixel_array = dicom_image.pixel_array
        shape = pixel_array.shape
        my_string = pixel_array.astype(np.uint8)
        sudhi = my_string
        print("\n\nEntered DICOM image data is:")
        message = str(my_string.tolist())
```

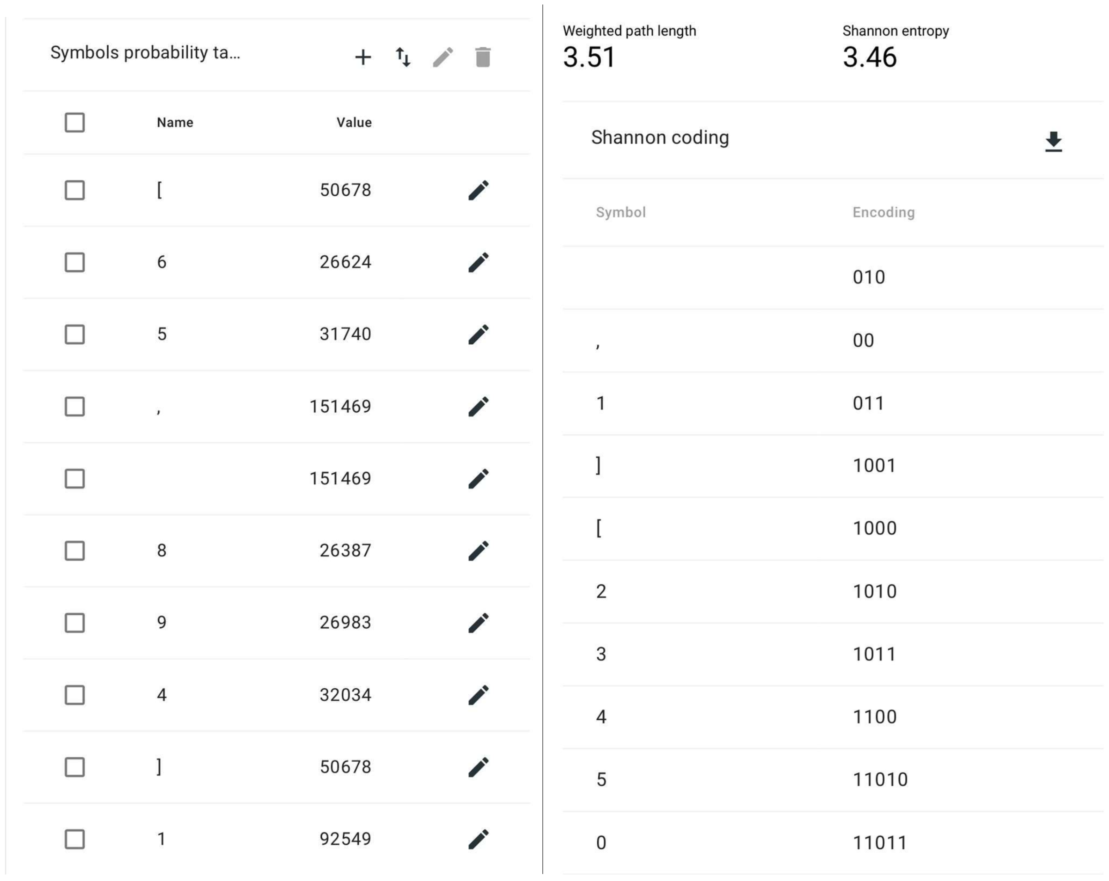
```
else:
    print("\n\nYou entered an invalid input")
    exit()

with open(os.path.join(results_folder, 'rawdata.txt'), 'w') as file:
    file.write(message)

c = {}

# Function to calculate probabilities and initialize the list
def create_list(message):
    list = dict(collections.Counter(message))
    total_count = sum(list.values())
    for key, value in list.items():
        probability = round(value / total_count, 4)
        print(key, ' :  Count: ', value, ' Probability: ', probability)
    list_sorted = sorted(list.items(), key=lambda k_v: (k_v[1], k_v[0]), reverse=True)
    final_list = []
    for key, value in list_sorted:
        final_list.append([key, value, ''])
    print("\n")
    print("Shannon Fano Process:")
    return final_list

# Function to divide the list in two parts as evenly as possible
def divide_list(list):
    total_sum = sum(i[1] for i in list)
    running_sum = 0
    min_diff = float('inf')
    split_index = -1

    # Find the index that minimizes the difference between two groups
    for i in range(len(list)):
        running_sum += list[i][1]
        diff = abs(running_sum - (total_sum - running_sum))
        if diff < min_diff:
            min_diff = diff
            split_index = i

    return list[:split_index+1], list[split_index+1:]

# Function to recursively label the symbols
def label_list(list):
    if len(list) == 1:
        return
        list1, list2 = divide_list(list)

        for i in list1:
            i[2] += '0'
            c[i[0]] = i[2]
        for i in list2:
            i[2] += '1'
            c[i[0]] = i[2]

        # Recursive labeling
        label_list(list1)
        label_list(list2)

        return c

    code = label_list(create_list(message))
    print("\nShannon's Encoded Code:")

    letter_binary = []
    for key, value in code.items():
        print(key, ' : ', value)
        letter_binary.append([key, value])

    symbol_stats = []
    with open(os.path.join(results_folder, 'symbolstats.txt'), 'w') as file:
        for key, value in code.items():
            probability = message.count(key) / len(message)
            file.write(f"Symbol: {key}, Probability: {probability:.6f}, Code: {value}\n")
            symbol_stats.append({'Symbol': key, 'Probability': probability, 'Code Length':
len(value)})

    with open(os.path.join(results_folder, 'compressed.txt'), 'w') as output:
        for a in message:
            for key, value in code.items():
                if key in a:
                    output.write(value)

    with open(os.path.join(results_folder, 'compressed.txt'), 'r') as output:
        bitstring = output.read()

    uncompressed_string = ""
    code = ""
    for digit in bitstring:
        code += digit
        pos = 0
    for letter in letter_binary:
        if code == letter[1]:
            uncompressed_string += letter_binary[pos][0]
            code = ""
        pos += 1

with open(os.path.join(results_folder, 'uncompressed.txt'), 'w') as file:
    file.write(uncompressed_string)

if h == 1 or h == 2:
    temp = re.findall(r'\d+', uncompressed_string)
    res = list(map(int, temp))
    res = np.array(res)
    res = res.astype(np.uint8)
    res = np.reshape(res, shape)
    print("\n\nObserve the shapes and input and output arrays are matching or not")
    print("Input image dimensions:", shape)
    print("Output image dimensions:", res.shape)
    data = Image.fromarray(res)
    data.save(os.path.join(results_folder, 'uncompressed.jpg'))
    if np.array_equal(sudhi, res):
        print("Success\n")

print("Symbol Statistics:")
entropy = 0
lavg = 0
efficiency = 0

symbols = []
counts = []
code_lengths = []

for stat in symbol_stats:
    symbols.append(stat['Symbol'])
    counts.append(stat['Probability'] * len(message))
    code_lengths.append(stat['Code Length'])

    p = stat['Probability']
    l = stat['Code Length']
    entropy += (p * math.log2(1/p))
    lavg += (p * l)

efficiency = entropy / lavg
```

```
    print(f"Entropy: {entropy:.4f}\nAverage CodeWordLength: {lavg:.4f}\nEfficiency:
{efficiency*100:.2f}%\n")

    # Calculate PSNR
    try:
        psnr_value = calculate_psnr(name, os.path.join(results_folder, 'uncompressed.jpg'))
        print(f"PSNR: {psnr_value:.2f}dB\n")
        if(psnr_value > 40):
            psnr = "Excellent quality, nearly imperceptible differences"
        elif((psnr_value > 30) and (psnr_value <= 40)):
            psnr = "Good quality with slight visible differences"
        elif((psnr_value > 20) and (psnr_value <= 30)):
            psnr = "Noticeable degradation in quality"
        elif(psnr_value <= 20):
            psnr = "Poor quality with significant differences"
        elif(psnr_value == float('inf')):
            psnr = "Exact Same Images!"
        print(psnr)
        print('\n')

    except ValueError as e:
        print(e)

    fig, ax1 = plt.subplots()

    color = 'tab:blue'
    ax1.set_xlabel('Symbols')
    ax1.set_ylabel('Count', color=color)
    ax1.bar(symbols, counts, color=color, alpha=0.6)
    ax1.tick_params(axis='y', labelcolor=color)

    ax2 = ax1.twinx()
    color = 'tab:red'
    ax2.set_ylabel('Code Length', color=color)
    ax2.plot(symbols, code_lengths, color=color, marker='o')
    ax2.tick_params(axis='y', labelcolor=color)

    fig.tight_layout()
    plt.title('Symbol Count and Code Length')
    plt.savefig(os.path.join(results_folder, 'symbol_count_code_length.png'))

    # Plot pie chart for entropy and average code word length
    labels = ['Entropy', 'Average Code Length']
    sizes = [entropy, lavg]
    colors = ['gold', 'lightcoral']
    explode = (0.1, 0)

    fig1, ax1 = plt.subplots()
    ax1.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%',
            shadow=True, startangle=140)
    ax1.axis('equal')

    plt.title('Ratio of Entropy and Average Code Length')
    plt.savefig(os.path.join(results_folder, 'entropy_avg_code_length_ratio.png'))

    print(f"Results saved in folder: {results_folder}")

print("\nProcess completed!\n\n")
```

CHAPTER 9

RESULT WITH GRAPH

✓ SYMBOL COUNT & CODEWORD MANUAL CALCULATION:



Symbols probability ta…

Name	Value
[	50678
6	26624
5	31740
,	151469
	151469
8	26387
9	26983
4	32034
]	50678
1	92549

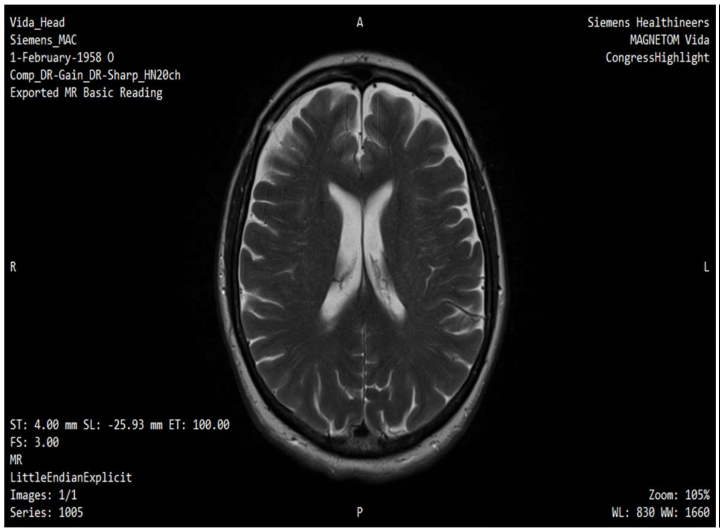
Weighted path length 3.51

Shannon entropy 3.46

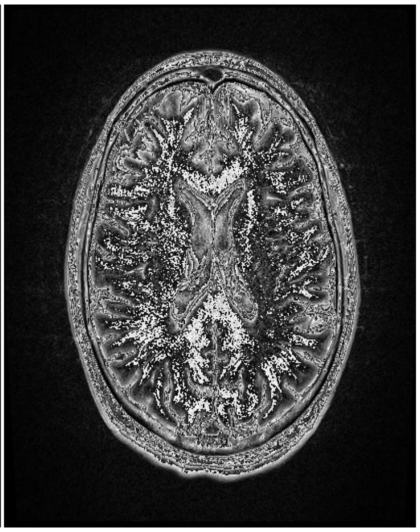
Shannon coding

Symbol	Encoding
	010
,	00
1	011
]	1001
[	1000
2	1010
3	1011
4	1100
5	11010
0	11011

Original Image Vs Shannon Fano Coded and Decoded JPG Image

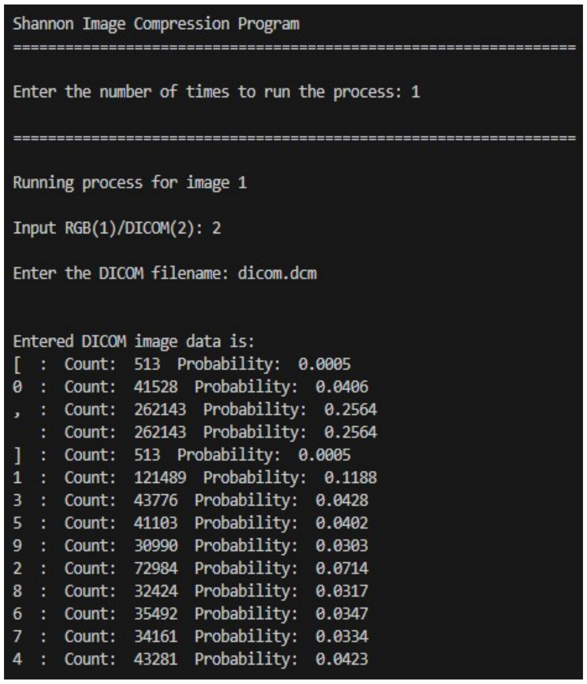


Original DICOM Image

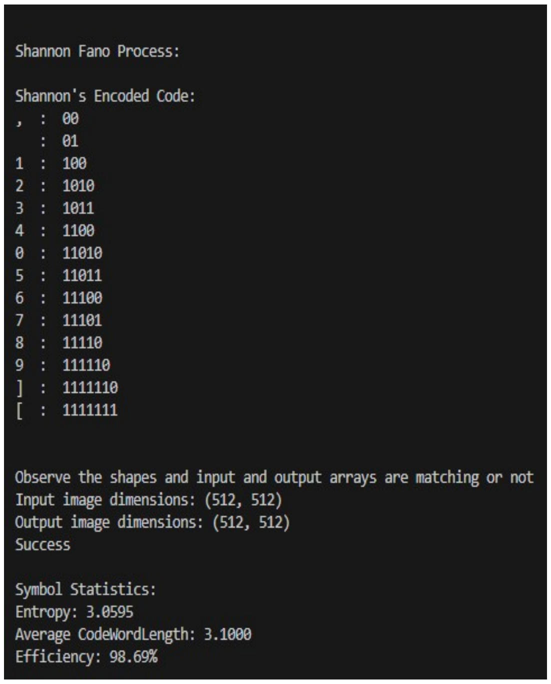


Decoded as JPEG after Encoding using Shannon-Fano

OUTPUT

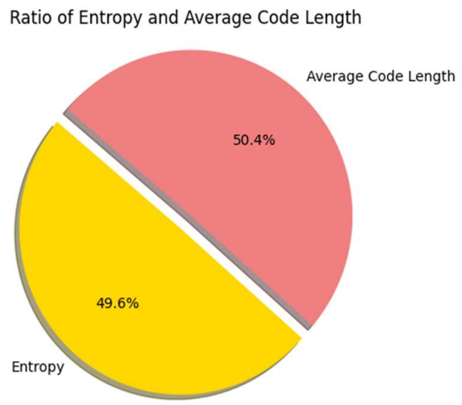


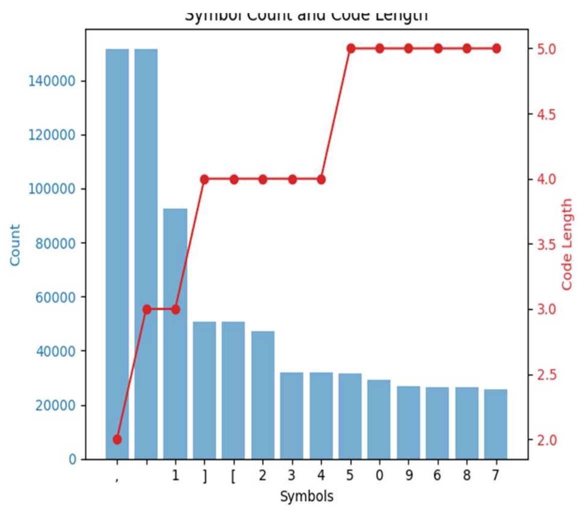
```
Shannon Image Compression Program
================================================================

Enter the number of times to run the process: 1

================================================================

Running process for image 1

Input RGB(1)/DICOM(2): 2

Enter the DICOM filename: dicom.dcm


Entered DICOM image data is:
[  :  Count:  513  Probability:  0.0005
0  :  Count:  41528  Probability:  0.0406
,  :  Count:  262143  Probability:  0.2564
   :  Count:  262143  Probability:  0.2564
]  :  Count:  513  Probability:  0.0005
1  :  Count:  121489  Probability:  0.1188
3  :  Count:  43776  Probability:  0.0428
5  :  Count:  41103  Probability:  0.0402
9  :  Count:  30990  Probability:  0.0303
2  :  Count:  72984  Probability:  0.0714
8  :  Count:  32424  Probability:  0.0317
6  :  Count:  35492  Probability:  0.0347
7  :  Count:  34161  Probability:  0.0334
4  :  Count:  43281  Probability:  0.0423
```



```
Shannon Fano Process:

Shannon's Encoded Code:
,  :  00
   :  01
1  :  100
2  :  1010
3  :  1011
4  :  1100
0  :  11010
5  :  11011
6  :  11100
7  :  11101
8  :  11110
9  :  111110
]  :  1111110
[  :  1111111


Observe the shapes and input and output arrays are matching or not
Input image dimensions: (512, 512)
Output image dimensions: (512, 512)
Success

Symbol Statistics:
Entropy: 3.0595
Average CodeWordLength: 3.1000
Efficiency: 98.69%
```

Charts And Visualizations of Various Metrices In Shannon Fano Coding





CHAPTER 10

CONCLUSION

In this project, we successfully implemented Shannon-Fano coding for LAB images, showcasing its effectiveness as a lossless compression technique. The analysis revealed substantial efficiency and compression while ensuring that the quality of the images remained intact

We have successfully analysed the results using graphs and charts and also generated the efficiency metrics

We have further extended our project for medical images, and futher in study exploring its capabilities and potential exploring further possibilities

CHAPTER 11

REFERENCES

- *https://www.geeksforgeeks.org/shannon-fano-algorithm-for-data-compression/*
- *https://en.wikipedia.org/wiki/Shannon%E2%80%93Fano_coding*
- *Reddy, M. & Akshaya, Koganti & Seles, R. & Dhivya, RA & Ravichandran, Kattur Soundarapandian. (2018). Image Compression using Shannon-Fano-Elias Coding and Run Length Encoding. 1-5. 10.1109/ICICCT.2018.8473120.*
- *Rahman, Md & Mohamed, Mohamed. (2019). Lossless Image Compression Techniques: A State-of-the-Art Survey. Symmetry. 11. 1274. 10.3390/sym11101274.*