IMAGE COMPRESSION

Deepak Somasekar	CB.EN.U4CCE21014
Harini Raj Akhtshya	CB.EN.U4CCE21026
Dharshini Muthu Bala	CB.EN.U4CCE21041
Renganathan	CB.EN.U4CCE21052

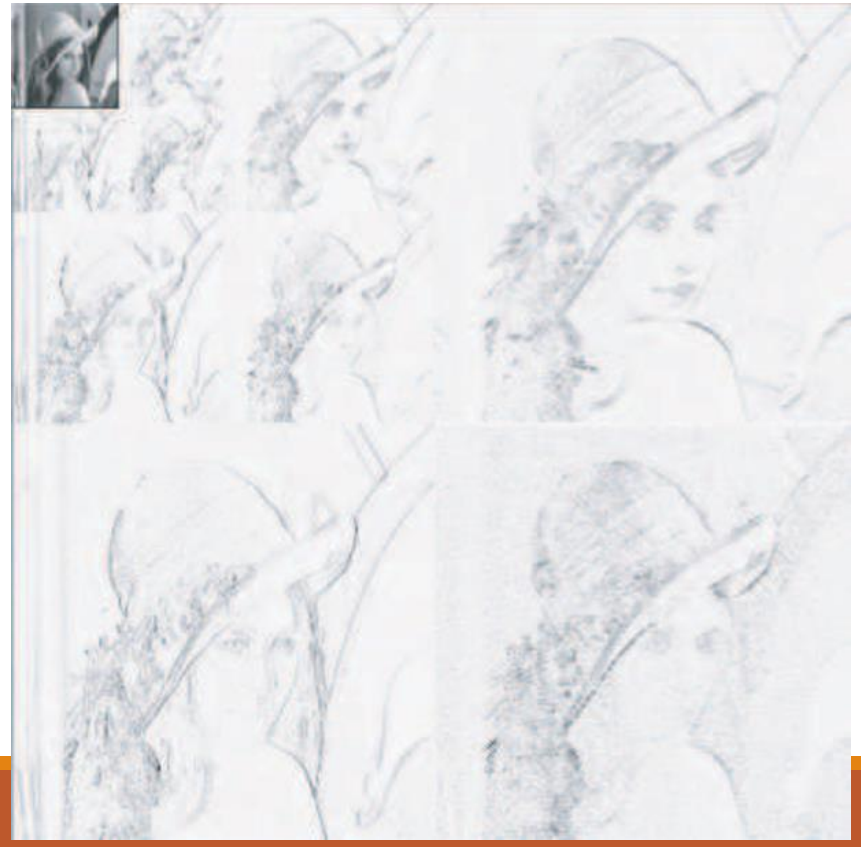
INTRODUCTION

What is Image Compression?

Reducing the file size of an image, while maintaining an acceptable level of quality.

Why Use Wavelets for Image Compression?

- ✓Images are represented at different resolutions or scales.
- ✓Better at capturing edges in the image.



Existing Technologies

Lossy Compression

JPEG (Joint Photographic Experts Group):

Uses **Discrete Cosine Transform (DCT)** to convert spatial image data into frequency components.

WebP:

Combines DCT with predictive coding.

HEIC (High-Efficiency Image Coding):

Uses **transform coding** to predict and encode pixel values efficiently.

Lossless Compression

PNG (Portable Network Graphics):

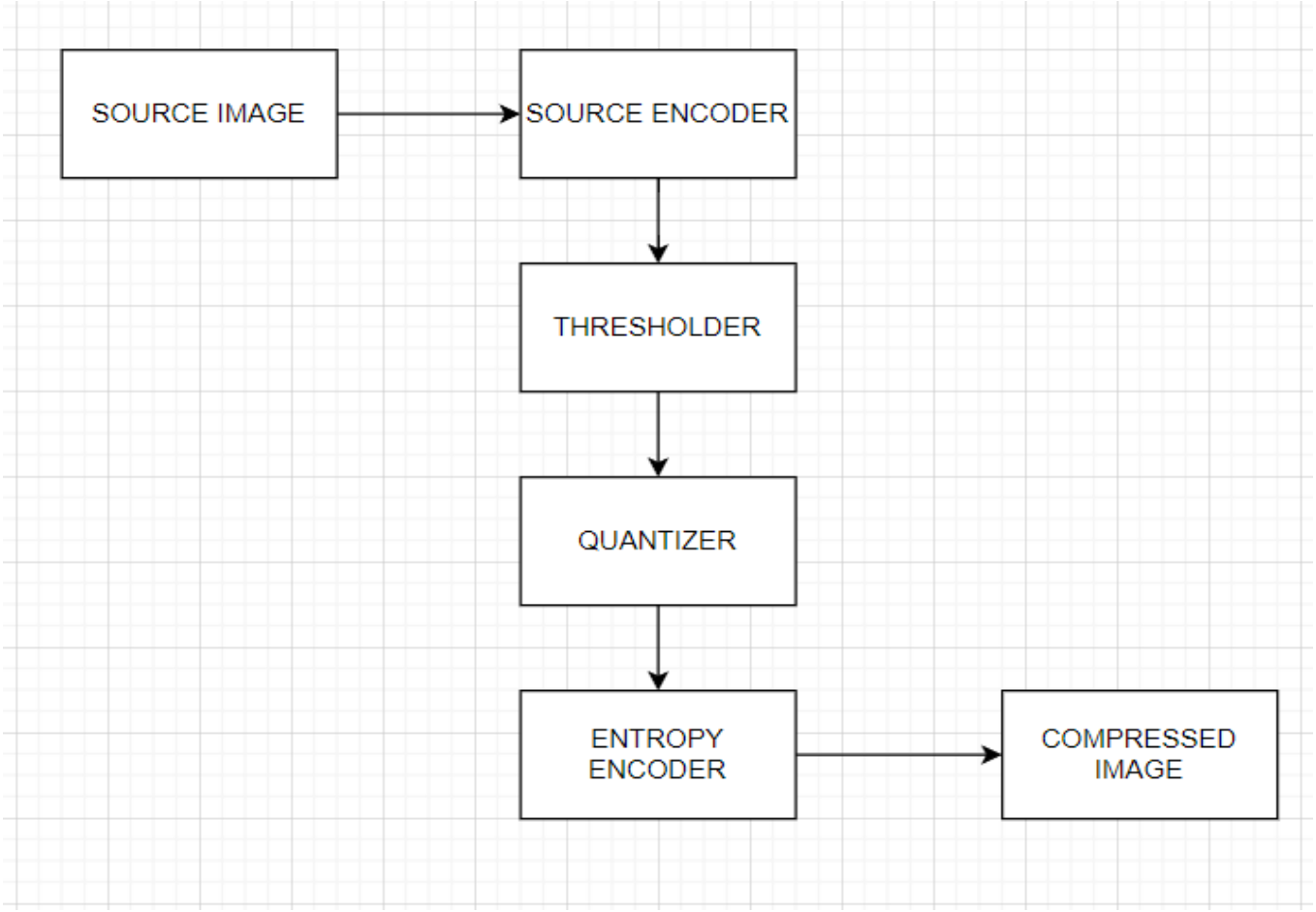
Uses **DEFLATE**, a combination of LZ77 algorithm and Huffman coding, to compress without losing data.

Deep Learning Models

Neural networks (e.g., convolutional neural networks, or CNNs) analyze image patterns and compress intelligently by predicting essential details.

Generative adversarial networks (GANs) reconstruct images using compact representations.

METHODOLOGY



```python
import sys, os, time, numpy, pywt
import matplotlib.pyplot as plt
from PIL import Image

def wavelet_transform(data, threshold):
    wavelet_type = 'haar'
    clean_coef = list()
    compose = list()

    cA2, cD2, cD1 = pywt.wavedec2(data, wavelet_type, level=2)
    clean_coef.append(cA2)
    clean_coef.append(cD2)

    for c in cD1:
        compose.append(numpy.where(((c<(-threshold)) | (c>threshold)), c, 0))
    clean_coef.append(tuple(compose))

    t = pywt.waverec2(clean_coef, wavelet_type)
    values = t.astype(int)
    return values

```

```python
def create_image(image, values, threshold):
    matrix = list()
    for value in values:
        row = list()
        for v in value:
            row.append((int(v), int(v), int(v)))
        matrix.append(row)

    width, height = image.size
    new_image = Image.new('RGB', (width, height))
    new = new_image.load()
    for w in range(width):
        for h in range(height):
            new[w, h] = matrix[h][w]

    image_name = str(threshold) + '.png'
    new_image.save(image_name)
    return new_image
```

```python
def grayscale(image):
    width, height = image.size
    pixels = image.load()

    for w in range(width):
        for h in range(height):
            r, g, b = pixels[w, h]
            gray = (r+g+b)//3
            pixels[w, h] = (gray, gray, gray)
    return image

def get_rows_values(image):
    width, height = image.size
    pixels = image.load()
    matrix = list()

    for j in range(height):
        row = list()
        for i in range(width):
            pixel_value = pixels[i, j][0]
            row.append(pixel_value)
        matrix.append(row)

    array = numpy.array(matrix)
    return array
```

```
67  def compress(image_path, threshold):
68      image = Image.open(image_path).convert('RGB')
69      image = grayscale(image)
70
71      data = get_rows_values(image)
72      values = wavelet_transform(data, threshold)
73
74      newimage = create_image(image, values, threshold)
75      return compressed_percentage(image_path, threshold)
76
77  def compressed_percentage(image_path, threshold):
78      original_size = os.path.getsize(image_path)
79      image_name = str(threshold) + '.png'
80      final_size = os.path.getsize(image_name)
81      percentage = 100 - (final_size*100)//float(original_size)
82      print ('Image compressed at %0.2f%%' % percentage)
83      return percentage
```

```python

def main():
    if len(sys.argv) > 1:
        image_path = sys.argv[1]

        time_list = list()
        percentages_list = list()
        thresholds_list = list()
        for threshold in range(0, 200, 20):
            start_time = time.time()
            compressed_percentage = compress(image_path, threshold)
            end_time = time.time()
            process_time = end_time - start_time
            time_list.append(process_time)
            percentages_list.append(compressed_percentage)
            thresholds_list.append(threshold)

        p = plt.plot(thresholds_list, percentages_list, 'bo-', label='Percentage')
        plt.legend(loc='upper left', numpoints=1)
        plt.ylabel('Percentage')
        plt.xlabel('Threshold value')
        plt.title('Percentage vs. Threshold value')
        plt.show()

        average_time = sum(time_list)//len(time_list)
        print ('The average time is', average_time)
    else:
        print ('Missing image path')

if __name__ == '__main__':
    main()
```

THANK YOU!