```
from google.colab import files
from IPython.display import Image
uploaded=files.upload()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[2], line 1
----> 1 from google.colab import files
      2 from IPython.display import Image
      3 uploaded=files.upload()

ModuleNotFoundError: No module named 'google.colab'
```

```
import matplotlib.pyplot as plt
import cv2
from google.colab.patches import cv2_imshow

image_path = 'bi.webp'

image=cv2.imread(image_path)
cv2_imshow(image)
```

```
import numpy as np
import cv2
from google.colab.patches import cv2_imshow  # Use this only in Google Colab

# Load the image correctly
image_path = 'bi.webp'  # Replace with the actual path
image = cv2.imread(image_path)

# Check if the image is loaded correctly
if image is None:
    raise FileNotFoundError(f"Error: Could not load the image. Check the path: {image_path}")

# Convert to grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Get height and width
height, width = gray_image.shape

# Compression using Run-Length Encoding (RLE)
compressed_image = []
for row in range(height):
    cur_pixel = None
    cur_run = 0

    for col in range(width):
        pixel = gray_image[row, col]
        if pixel == cur_pixel:
            cur_run += 1
        else:
            if cur_pixel is not None:  # Append only if it's not the first pixel
                compressed_image.append((cur_pixel, cur_run))
            cur_pixel = pixel
            cur_run = 1
```

```
        # Append last run in the row
        compressed_image.append((cur_pixel, cur_run))

    # Decompression
    decompressed_image = np.zeros((height, width), dtype=np.uint8)

    row, col = 0, 0
    for pixel, run_length in compressed_image:
        decompressed_image[row, col:col + run_length] = pixel
        col += run_length

        if col >= width:
            row += 1
            col = 0  # Reset column after moving to a new row

    # Display images
    print("Original Image")
    cv2_imshow(gray_image)  # Display the original grayscale image

    print("Decompressed Image")
    cv2_imshow(decompressed_image)  # Display the decompressed image
```

```
Original Image
```

```
Decompressed Image
```

```
    import cv2
    import numpy as np
    from google.colab.patches import cv2_imshow  # Required for displaying images in Colab

    # Load the image correctly
    image_path = 'bi.webp'  # Ensure the file is uploaded to Colab
    image = cv2.imread(image_path)

    # Check if the image is loaded properly
    if image is None:
        print(f"Error: Unable to load image at {image_path}")
        exit()

    # Convert to grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
# Function to compress a region using DCT
def compress_region(region):
    region_height, region_width = region.shape

    # Apply DCT to the region
    dct_region = cv2.dct(region.astype(np.float32))

    # Set low-frequency coefficients to zero for compression
    dct_region[region_height//8:, region_width//8:] = 0

    # Apply inverse DCT
    idct_region = cv2.idct(dct_region)

    return idct_region

# Create a copy of the image to store the compressed version
compressed_image = gray_image.copy()

# Iterate over 8x8 blocks in the image
height, width = gray_image.shape
for i in range(0, height, 8):
    for j in range(0, width, 8):
        block = gray_image[i:i+8, j:j+8]  # Extract 8x8 block
        compressed_block = compress_region(block)  # Apply DCT compression
        compressed_image[i:i+8, j:j+8] = compressed_block  # Store compressed block

# Display the original and compressed images using cv2_imshow() in Colab
print("Original Image:")
cv2_imshow(gray_image)

print("Compressed Image:")
cv2_imshow(compressed_image)
```

```
Original Image:
```

```
Compressed Image:
```

```
import numpy as np
import cv2
from google.colab.patches import cv2_imshow  # Use this in Google Colab
```

```
def quantize(image, levels):
    flat_image = image.flatten()  # Flatten the image into a 1D array
    min_val = flat_image.min()
    max_val = flat_image.max()

    # Divide the range of pixel values into equally spaced intervals
    interval = (max_val - min_val) / levels
    intervals = [min_val + i * interval for i in range(levels + 1)]

    # Compute mean values of each interval
    means = [(intervals[i] + intervals[i + 1]) / 2 for i in range(levels)]

    # Replace pixel values with the mean value of their interval
    quantized_image = np.array([means[np.searchsorted(intervals, val) - 1] for val in flat_image])

    # Reshape the quantized image back to the original shape
    return quantized_image.reshape(image.shape).astype(np.uint8)

# Load a grayscale image
image_path = 'bi.webp'  # Replace with your actual image path
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

# Apply quantization with 16 levels
quantized_image = quantize(image, 16)

# Display the original and quantized images
print("Original Image:")
cv2_imshow(image)

print("Compressed Image:")
cv2_imshow(quantized_image)
```

```
Original Image:
```

```
Compressed Image:
```