```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files
from google.colab.patches import cv2_imshow

uploaded = files.upload()


filenames = list(uploaded.keys())
if len(filenames) < 2:
    raise Exception("Please upload 2 images: one grayscale and one color image.")

gray_path = filenames[0]
color_path = filenames[1]

gray_img = cv2.imread(gray_path, cv2.IMREAD_GRAYSCALE)
color_img = cv2.imread(color_path, cv2.IMREAD_COLOR)
```

Choose Files 2 files

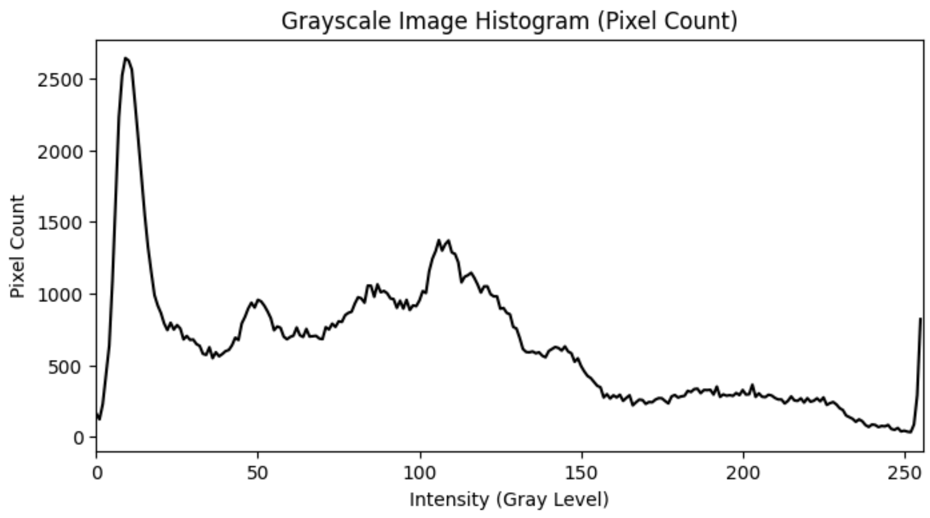
- **gray_image.jpg**(image/jpeg) - 62148 bytes, last modified: 2/6/2025 - 100% done
- **logo.jpg**(image/jpeg) - 488402 bytes, last modified: 1/23/2025 - 100% done

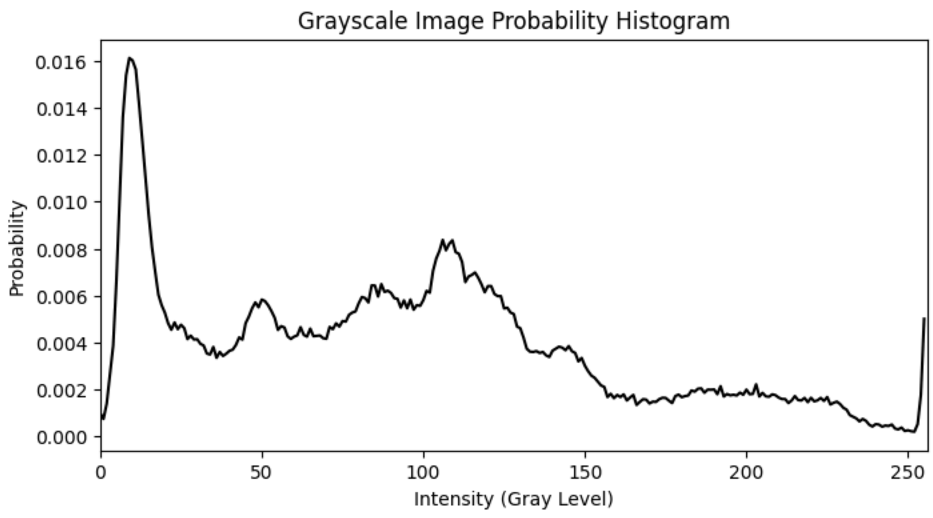
```
Saving gray_image.jpg to gray_image (4).jpg
```

```python
def compute_histogram(image, color=False):
    """
    Compute and display histogram for a grayscale or color image.
    M1: x-axis represents intensity levels (0-255), y-axis is the pixel count.
    """
    plt.figure(figsize=(8, 4))
    if color:
        # For color image, process each channel (B, G, R)
        channels = cv2.split(image)
        colors = ('b', 'g', 'r')
        for i, col in enumerate(colors):
            hist = cv2.calcHist([channels[i]], [0], None, [256], [0, 256])
            plt.plot(hist, color=col, label=f'{col.upper()} channel')
        plt.title("Color Image Histogram (Pixel Count)")
        plt.legend()
    else:
        hist = cv2.calcHist([image], [0], None, [256], [0, 256])
        plt.plot(hist, color='black')
        plt.title("Grayscale Image Histogram (Pixel Count)")
    plt.xlabel("Intensity (Gray Level)")
    plt.ylabel("Pixel Count")
    plt.xlim([0, 256])
    plt.show()

def probability_histogram(image):
    """
    Compute and display probability histogram.
    M2: x-axis represents intensity levels, y-axis represents the probability of each level.
    """
    plt.figure(figsize=(8, 4))
    hist = cv2.calcHist([image], [0], None, [256], [0, 256])
    hist = hist / hist.sum()  # Normalize to get probabilities
    plt.plot(hist, color='black')
    plt.title("Grayscale Image Probability Histogram")
    plt.xlabel("Intensity (Gray Level)")
    plt.ylabel("Probability")
    plt.xlim([0, 256])
    plt.show()

def apply_histogram_equalization_gray(image):
    """
    Enhance contrast of a grayscale image using histogram equalization.
    """
    return cv2.equalizeHist(image)

def apply_histogram_equalization_color(image):
    """
    Enhance contrast of a color image by applying histogram equalization on each channel.
    """
    channels = cv2.split(image)
    eq_channels = [cv2.equalizeHist(ch) for ch in channels]
    eq_image = cv2.merge(eq_channels)
    return eq_image
```

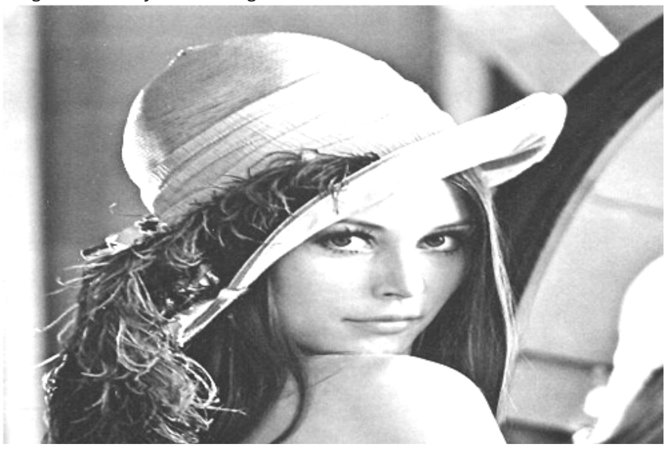
```
print("Original Grayscale Image:")
cv2_imshow(gray_img)

# (a) Compute & display histogram (Method 1: Pixel Count)
compute_histogram(gray_img, color=False)

# (b) Compute & display probability histogram (Method 2)
probability_histogram(gray_img)

# (c) Simulate Bright and Dark images by adjusting brightness
bright_img = cv2.convertScaleAbs(gray_img, alpha=1.5, beta=50)   # brighten
dark_img   = cv2.convertScaleAbs(gray_img, alpha=0.5, beta=-50)  # darken

print("Brightened Grayscale Image:")
cv2_imshow(bright_img)
compute_histogram(bright_img, color=False)

print("Darkened Grayscale Image:")
cv2_imshow(dark_img)
compute_histogram(dark_img, color=False)

equalized_gray = apply_histogram_equalization_gray(gray_img)
print("Histogram Equalized Grayscale Image:")
cv2_imshow(equalized_gray)
compute_histogram(equalized_gray, color=False)
```

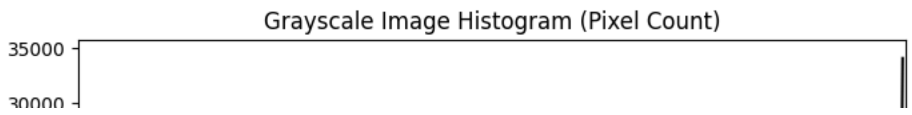
```
Original Grayscale Image:
```

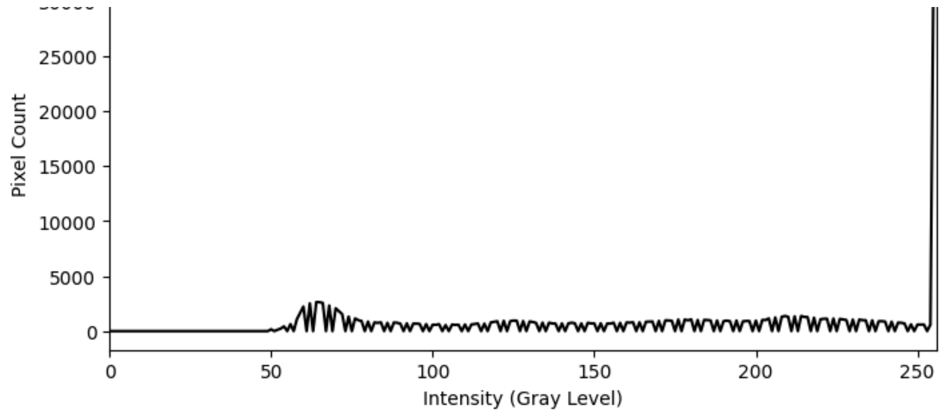




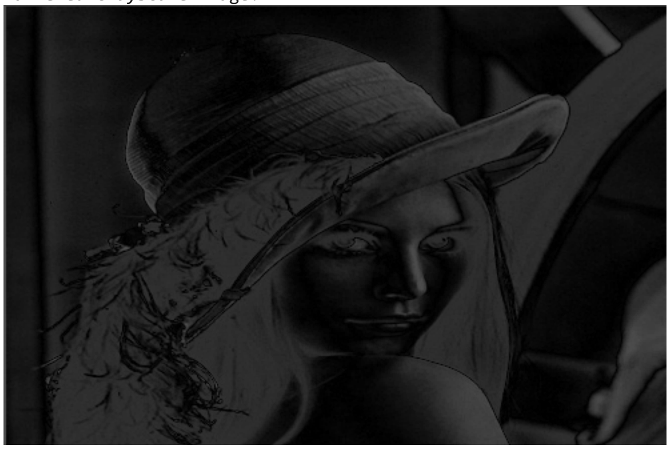
```
Brightened Grayscale Image:
```

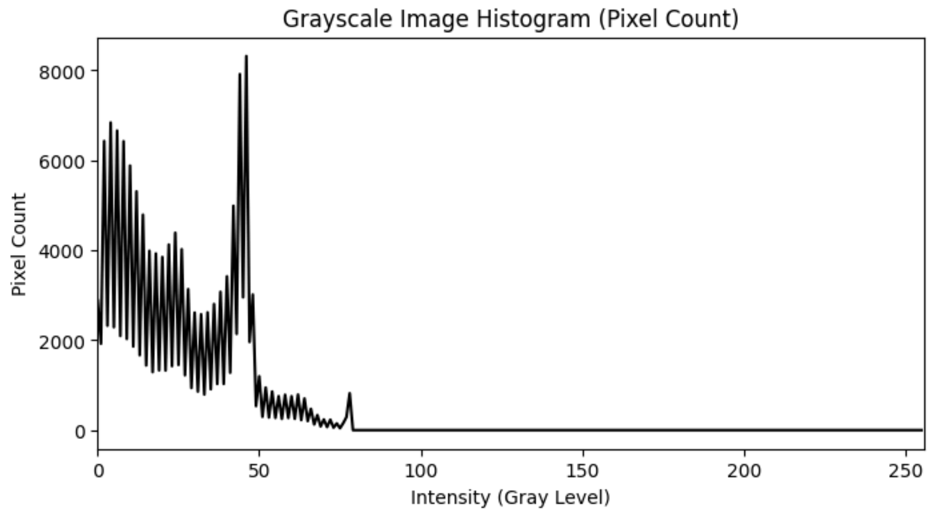




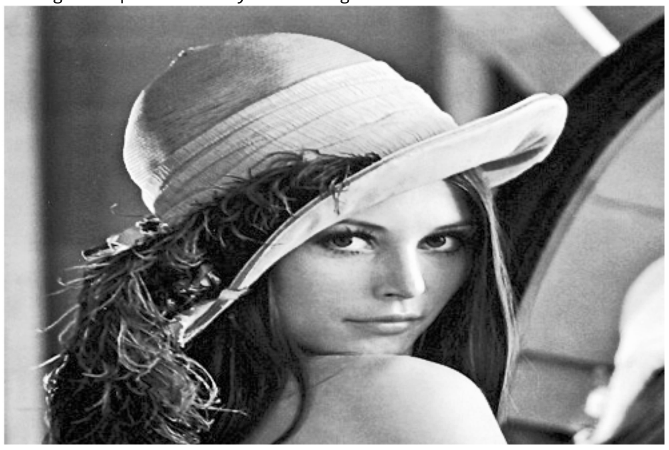


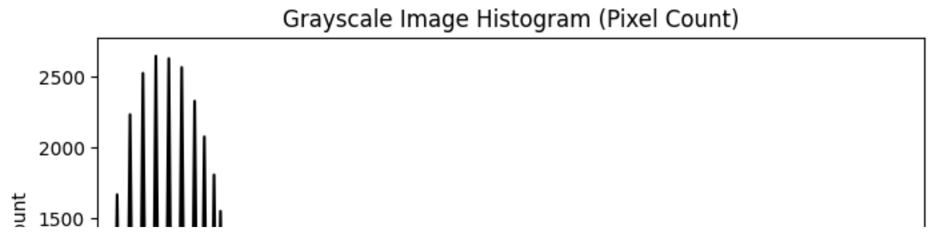
```
Darkened Grayscale Image:
```

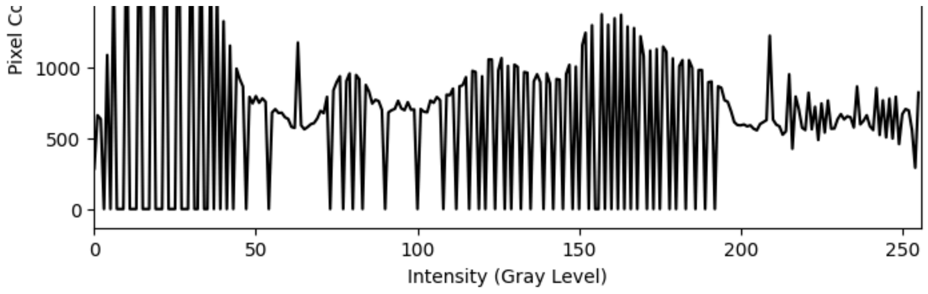




```
Histogram Equalized Grayscale Image:
```



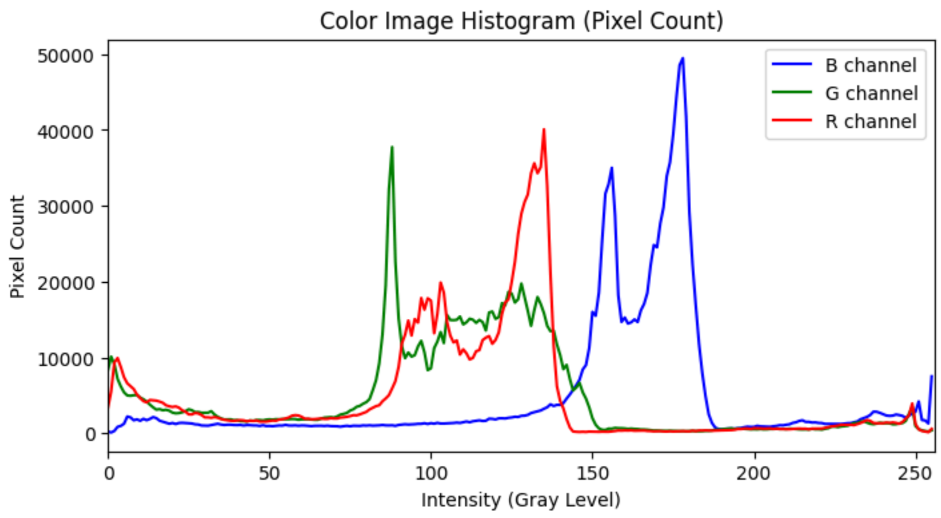



Pixel Co
1000
500
0
0
50
100
150
200
250
Intensity (Gray Level)

```
print("Original Color Image:")
cv2_imshow(color_img)

compute_histogram(color_img, color=True)

color_to_gray = cv2.cvtColor(color_img, cv2.COLOR_BGR2GRAY)
print("Color Image Converted to Grayscale:")
cv2_imshow(color_to_gray)
compute_histogram(color_to_gray, color=False)
probability_histogram(color_to_gray)

equalized_color = apply_histogram_equalization_color(color_img)
print("Histogram Equalized Color Image:")
cv2_imshow(equalized_color)
compute_histogram(equalized_color, color=True)
```

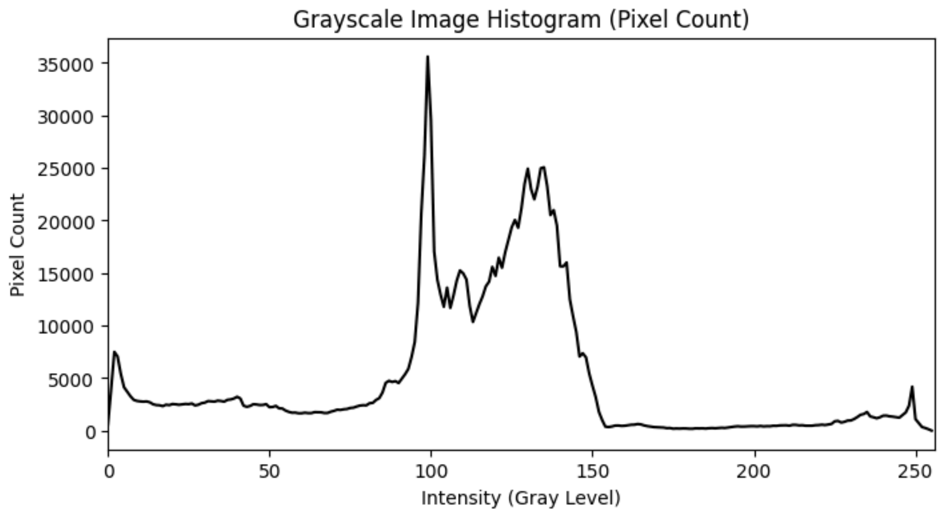
```
Original Color Image:
```

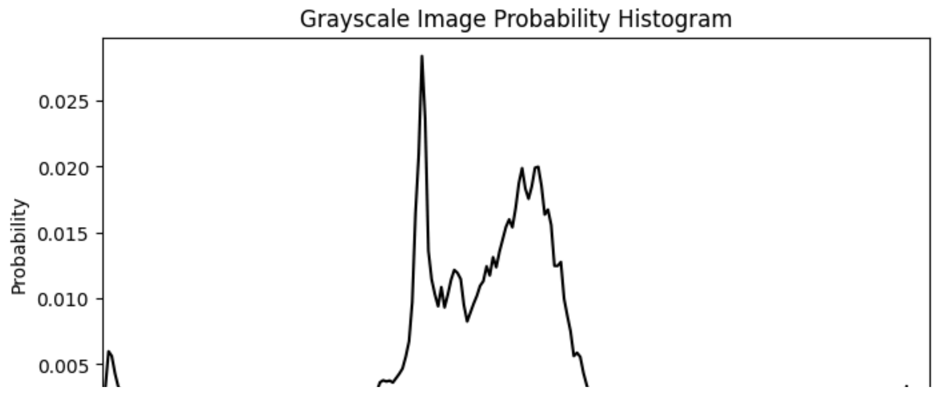
BITSCRUNCH
UNLEASHNFT V2
SDK

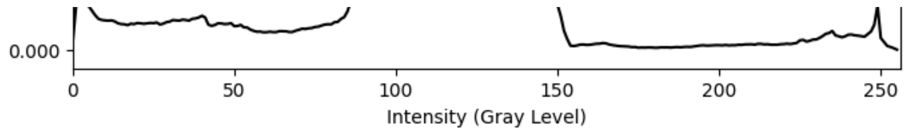


```
Color Image Converted to Grayscale:
```

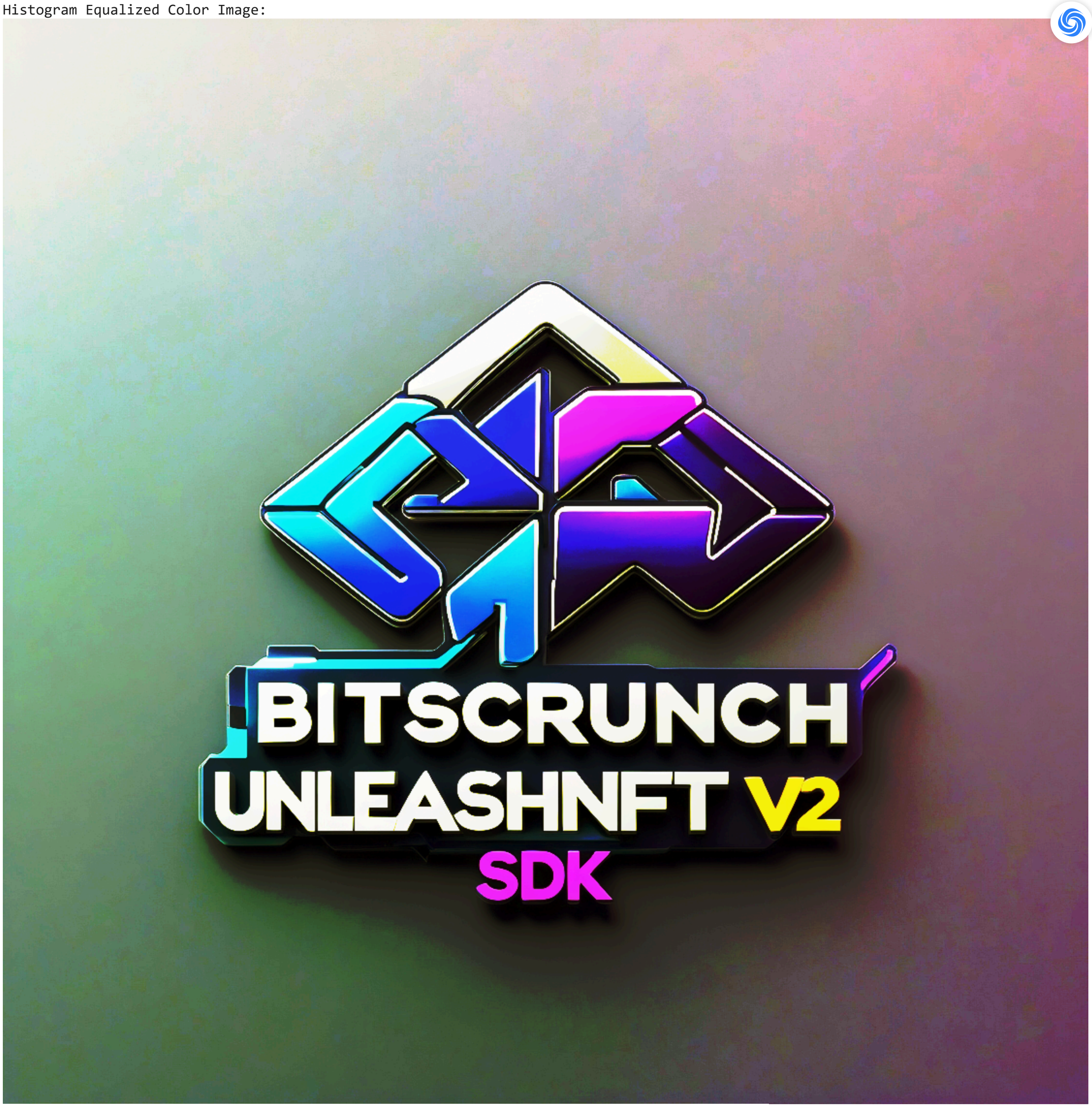
BITSCRUNCH
UNLEASHNFT V2
SDK

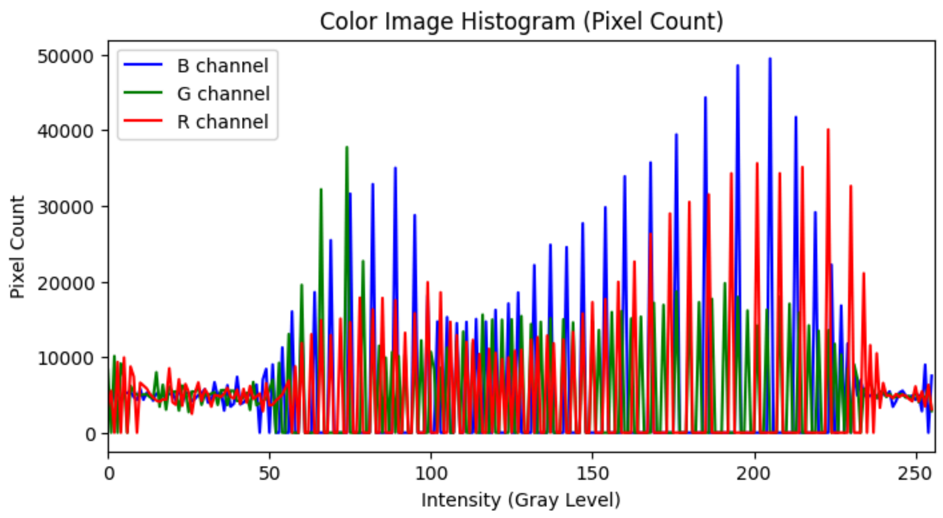

Grayscale Image Histogram (Pixel Count)
Pixel Count
Intensity (Gray Level)
0
5000
10000
15000
20000
25000
30000
35000
0
50
100
150
200
250


Grayscale Image Probability Histogram
Probability
0.005
0.010
0.015
0.020
0.025



```
Histogram Equalized Color Image:
```


BITSCRUNCH
UNLEASHNFT V2
SDK


Color Image Histogram (Pixel Count)
B channel
G channel
R channel
Pixel Count
Intensity (Gray Level)