Group Member

65103103	นายณัฐวุฒิ จิระพันธ์
65121154	นางสาวชญาดา เอียดทองใส
65125056	นายธีรนัย สังเทระ
65132185	นางสาวเกวลิน เศวตะดุล

Assignment - SOM Based Image Quantization

1.ฟังก์ชันคำนวณค่า MSE (Mean Square Error):

```
def MSE(image_a, image_b):
    w, h, c = image_a.shape
    mse = np.sum((image_a - image_b) ** 2) / (w * h * c)
    return mse
```

- ฟังก์ชันนี้ใช้คำนวณค่า MSE ระหว่างภาพสองภาพ คือ image_a และ image_b
- ค่า MSE เป็นตัวชี้วัดทั่วไปที่ใช้วัดความแตกต่างระหว่างภาพต้นฉบับและภาพที่ผ่านการประมวลผล โดยคำนวณจากการรวมผลต่างของค่าพิกเซลที่ยกกำลังสองในแต่ละจุด แล้วหารด้วยจำนวนพิกเซลทั้งหมดรวมทั้งช่องสี (color channels)

2.การสร้างพาเลตสีด้วย Self-Organizing Map (SOM):

```
def ColourPaletteGeneration(image_in, epochs=10, initial_learn_rate=0.1):
    w, h, c = image_in.shape
    codebook = random.rand(8, 8, 3) * 255  # เริ่มต้น codebook ด้วยค่าสีแบบสุ่มในช่วง [0, 255]

    learn_rate = initial_learn_rate
    for epoch in range(epochs):
        for i in range(w * h):
            # เลือกพิกเซลแบบสุ่ม
            randomPixelRow = random.randint(0, h)
            randomPixelCol = random.randint(0, w)
            currentPixel = image_in[randomPixelRow, randomPixelCol, :]

            # ค้นหา codeword ที่ใกล้เคียงที่สุด
            minDist = np.inf
            minIndex_r, minIndex_c = 0, 0
            for r in range(8):
                for c in range(8):
                    dist = np.linalg.norm(currentPixel - codebook[r, c, :])
                    if dist < minDist:
                        minDist = dist
                        minIndex_r, minIndex_c = r, c
```

```
            # อัปเดตน้ำหนักของ codeword ที่ใกล้เคียงที่สุด
            codebook[minIndex_r, minIndex_c, :] += learn_rate * (currentPixel - codebook[minIndex_r,
minIndex_c, :])

        # ลดค่า learning rate ลง
        learn_rate *= 0.99  # ลด learning rate เมื่อเวลาผ่านไป

    return codebook
```

- ฟังก์ชันนี้ใช้อัลกอริธึม Self-Organizing Map (SOM) เพื่อสร้างพาเลตสีสำหรับภาพที่นำเข้า (image_in)
- พาเลตสีหรือ codebook ถูกเริ่มต้นด้วยตารางขนาด 8x8 ของสีสุ่ม
- ในแต่ละ epoch อัลกอริธึมจะสุ่มเลือกพิกเซลจากภาพต้นฉบับแล้วหาสีใน codebook ที่ใกล้เคียงที่สุด (จากระยะ Euclidean)
- สีที่ใกล้เคียงที่สุดใน codebook จะถูกปรับให้ใกล้เคียงกับสีของพิกเซลที่เลือกมากขึ้นโดยใช้อัตราการเรียนรู้ (learning rate)
- ค่า learning rate จะลดลงเล็กน้อยหลังจากแต่ละ epoch เพื่อให้การปรับแต่งละเอียดมากขึ้น

3.การแมปพิกเซลไปยังภาพที่ถูก Quantized:

```
def PixelMapping(image_in, codebook):
    w, h, c = image_in.shape
    image_out = np.zeros((w, h, c), dtype=np.uint8)

    for r in range(h):
        for c in range(w):
            currentPixel = image_in[r, c, :]

            # ค้นหา codeword ที่ใกล้เคียงที่สุด
            minDist = np.inf
            minIndex_r, minIndex_c = 0, 0
            for i in range(8):
                for j in range(8):
                    dist = np.linalg.norm(currentPixel - codebook[i, j, :])
                    if dist < minDist:
                        minDist = dist
                        minIndex_r, minIndex_c = i, j

            image_out[r, c, :] = codebook[minIndex_r, minIndex_c, :]

    return image_out
```

- ฟังก์ชันนี้ใช้ในการแมปพิกเซลในภาพต้นฉบับให้ตรงกับสีที่ใกล้เคียงที่สุดในพาเลตสีที่สร้างขึ้น (codebook) เพื่อสร้างภาพที่ถูก Quantized
- ผลลัพธ์คือภาพที่ใช้เฉพาะสีที่มีอยู่ใน codebook เท่านั้น ทำให้จำนวนสีในภาพลดลงอย่างมาก

4.การใช้งาน:

```
if __name__ == "__main__":
    # โหลดภาพ
    image_a = cv2.imread('baboon.jpeg')

    # สร้างพาเลตสีด้วย SOM
    palette = ColourPaletteGeneration(image_a, epochs=10, initial_learn_rate=0.1)

    # ทำการ Quantize ภาพ
    quantized_image = PixelMapping(image_a, palette)

    # คำนวณค่า MSE ระหว่างภาพต้นฉบับและภาพที่ถูก Quantized
    mse_value = MSE(image_a, quantized_image)
    print(f'MSE: {mse_value}')

    # บันทึกภาพที่ถูก Quantized
    cv2.imwrite('Quantized_Baboon.jpeg', quantized_image)
```

- โปรแกรมโหลดภาพ 'baboon.jpeg' และนำไปประมวลผลผ่าน SOM เพื่อสร้างภาพที่ถูก Quantized
- ค่า MSE ระหว่างภาพต้นฉบับและภาพที่ถูก Quantized จะถูกคำนวณและแสดงผล
- ภาพที่ถูก Quantized จะถูกบันทึกเป็นไฟล์ 'Quantized_Baboon.jpeg'

ผลลัพธ์และการตีความ

ค่า MSE:

ได้ค่า MSE ที่ 59.39032098765432

ค่า MSE ที่ต่ำกว่าจะบ่งบอกว่าภาพที่ถูก Quantized นั้นใกล้เคียงกับภาพต้นฉบับมากขึ้น ในขณะที่ค่า MSE ที่สูงกว่าจะบ่งบอกถึงความแตกต่างที่มากขึ้น ค่า MSE นี้ขึ้นอยู่กับปัจจัยต่าง ๆ เช่น จำนวน epoch, learning rate และลักษณะของภาพ

คุณภาพภาพที่ได้:

ภาพที่ถูก Quantized (Quantized_Baboon.jpeg) จะมีจำนวนน้อยลงของสีที่ใช้อยู่ในภาพ เมื่อเทียบกับภาพต้นฉบับเนื่องจากมันใช้เพียงแค่ 64 สีจากพาเลตขนาด 8x8 เท่านั้น
คุณภาพของการทำ Quantization ขึ้นอยู่กับว่า SOM สามารถปรับ codebook ให้เข้ากับสีในภาพต้นฉบับได้ดีเพียงใด

สรุป

โค้ดนี้ใช้วิธีการ SOM เพื่อการลดจำนวนสีในภาพ หรือที่เรียกว่า Image Quantization กระบวนการนี้ช่วยลดจำนวนสีในภาพในขณะที่พยายามรักษาคุณภาพของภาพไว้มากที่สุด ค่า MSE เป็นตัวชี้วัดเชิงปริมาณที่ใช้วัดความแตกต่างระหว่างภาพต้นฉบับและภาพที่ถูก Quantized อัลกอริธึม SOM สามารถปรับแต่งได้โดยการปรับจำนวน epoch และค่า learning rate เพื่อผลลัพธ์ที่ดีที่สุด