МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота № 2

з дисципліни «Теорія Інформації»
Варіант 31 (1)

Виконав:
студент групи КН-213
Ярмусь Віталій
Викладач:
Косаревич Р. Я.

Львів – 2020 р.

Тема: ЕКСПЕРИМЕНТАЛЬНЕ ВИЗНАЧЕННЯЕНТРОПІЇ ПОВІДОМЛЕННЯ

Мета роботи: Вивчення властивостей ентропії як кількісної міри інформації.

Хід роботи

1) Текст програми

```
static void Main(string[] args)
{
    Bitmap bmp = new Bitmap("C:\\Users\\Vitalii\\OneDrive\\University\\3 семестр\\Теорія Інформації\\2\\assets\\asmund-gimre-7SjmvQgBSKY-unsplash.jpg");
    Console.WriteLine($"Entropy of image: {FindEntropyOfImage(bmp)}");

    Bitmap sampled2 = new Bitmap(bmp, new Size(bmp.Width / 2, bmp.Height / 2));
    sampled2.Save("C:\\Users\\Vitalii\\OneDrive\\University\\3 семестр\\Теорія Інформації\\2\\assets\\sampled2.jpg", ImageFormat.Jpeg);
    Console.WriteLine($"Entropy of sampled 2 image: {FindEntropyOfImage(sampled2)}");

    Bitmap sampled4 = new Bitmap(bmp, new Size(bmp.Width / 4, bmp.Height / 4));
    sampled4.Save("C:\\Users\\Vitalii\\OneDrive\\University\\3 семестр\\Теорія Інформації\\2\\assets\\sampled4.jpg", ImageFormat.Jpeg);
    Console.WriteLine($"Entropy of sampled 4 image: {FindEntropyOfImage(sampled4)}\n");

    var qu = new ColorImageQuantizer(new MedianCutQuantizer());

    quantize(qu, sampled2, 8);
    quantize(qu, sampled2, 16);
    quantize(qu, sampled2, 64);

    restore(sampled2, bmp, 2);
    restore(sampled4, bmp, 4);
}

2 references
public static void restore(Bitmap sampled, Bitmap bmp, int n)
{
    Console.WriteLine("\n-----------------------------------------------------------------------------------------------------------\n");
    var restoredBicubic = EnlargeImage(sampled, n, InterpolationMode.Bicubic);
    restoredBicubic.Save($"C:\\Users\\Vitalii\\OneDrive\\University\\3 семестр\\Теорія Інформації\\2\\assets\\restored_{n}_bicubic.jpg", ImageFormat.Jpeg);
    Console.WriteLine($"Entropy of restored_{n}_bicubic: {FindEntropyOfImage(restoredBicubic)}");
    Console.WriteLine($"Relative entropy: {FindRelativeEntropyOfTwoImages(bmp, restoredBicubic)}\n");

    var restoredBilinear = EnlargeImage(sampled, n, InterpolationMode.Bilinear);
    restoredBilinear.Save($"C:\\Users\\Vitalii\\OneDrive\\University\\3 семестр\\Теорія Інформації\\2\\assets\\restored_{n}_bilibiar.jpg", ImageFormat.Jpeg);
    Console.WriteLine($"Entropy of restored_{n}_bilinear: {FindEntropyOfImage(restoredBilinear)}");
    Console.WriteLine($"Relative entropy: {FindRelativeEntropyOfTwoImages(bmp, restoredBilinear)}\n");
}

3 references
public static void quantize(ColorImageQuantizer qu, Bitmap bmp, int colors)
{
    Console.WriteLine("\n-----------------------------------------------------------------------------------------------------------\n");
    var quantized = qu.ReduceColors(bmp, colors);
    quantized.Save($"C:\\Users\\Vitalii\\OneDrive\\University\\3 семестр\\Теорія Інформації\\2\\assets\\quntized_{colors}.jpg", ImageFormat.Jpeg);
    Console.WriteLine($"Entropy of quntized_{colors}: {FindEntropyOfImage(quantized)}");
    Console.WriteLine($"Relative entropy: {FindRelativeEntropyOfTwoImages(bmp, quantized)}\n");
}
```

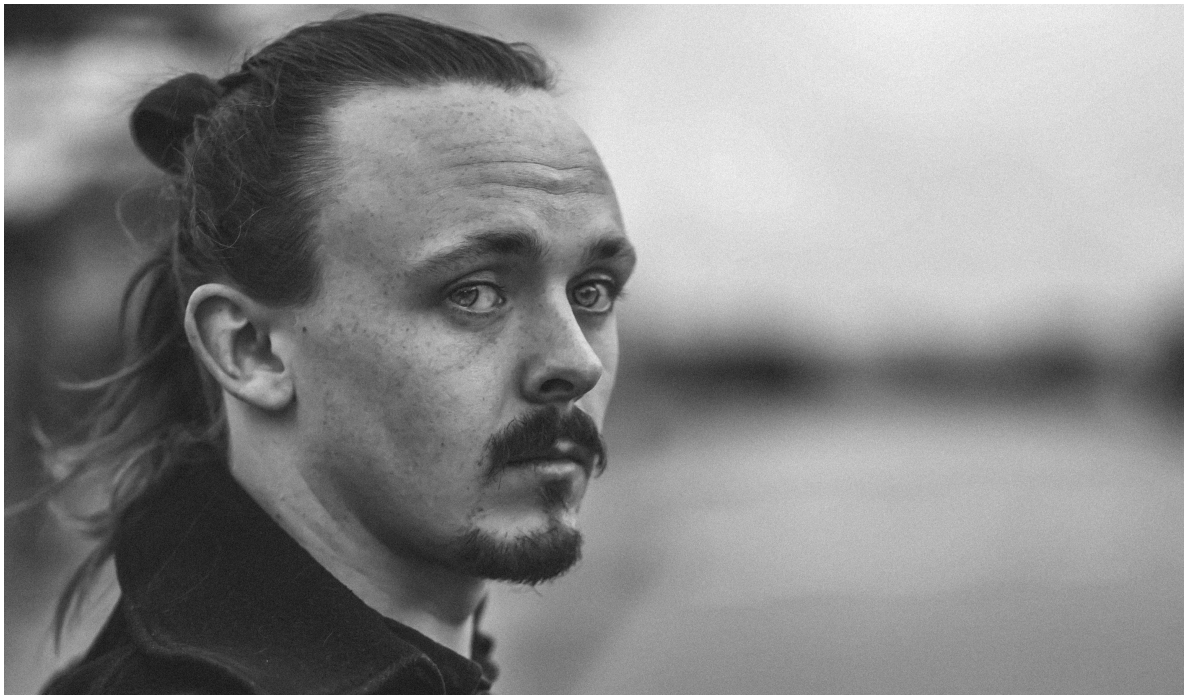
```
2 references
public static Bitmap EnlargeImage(Bitmap original, int scale, InterpolationMode interpolationMode)
{
    Bitmap newimg = new Bitmap(original.Width * scale, original.Height * scale);

    using (Graphics g = Graphics.FromImage(newimg))
    {
        g.InterpolationMode = interpolationMode;

        g.DrawImage(original, new Rectangle(Point.Empty, newimg.Size));
    }

    return newimg;
}

1 reference
public static double FindEntropy(double[] arr)
{
    double res = 0;
    foreach(var i in arr)
    {
        res += i * Math.Log2(i);
    }
    return -res;
}
```

```
3 references
public static double FindRelativeEntropyOfTwoImages(Bitmap img1, Bitmap img2)
{
    var numOfPixels = img1.Width * img1.Height;
    var pixelsMap1 = new Dictionary<byte, long>();
    var pixelsMap2 = new Dictionary<byte, long>();
    GroupPixels(img1, pixelsMap1);
    GroupPixels(img2, pixelsMap2);

    var probability1 = new Dictionary<byte, double>();
    foreach (KeyValuePair<byte, long> entry in pixelsMap1)
    {
        probability1.Add(entry.Key, entry.Value / (double)numOfPixels);
    }

    var probability2 = new Dictionary<byte, double>();
    foreach (KeyValuePair<byte, long> entry in pixelsMap2)
    {
        probability2.Add(entry.Key, entry.Value / (double)numOfPixels);
    }

    double relativeProbability = 0;
    for (byte i = 0; i <= 255; i++)
    {
        var px = probability1.GetValueOrDefault(i, 0);
        var qx = probability2.GetValueOrDefault(i, 0);
        if(px == 0)
        {
            continue;
        }
        if(qx == 0)
        {
            relativeProbability = Double.PositiveInfinity;
            break;
        }

        relativeProbability += px * Math.Log2(px / qx);
        if(i == 255)
        {
            break;
        }
    }
    return relativeProbability;
}
```

```
6 references
public static double FindEntropyOfImage(Bitmap bmp)
{
    var numOfPixels = bmp.Width * bmp.Height;
    var pixelsMap = new Dictionary<byte, long>();
    GroupPixels(bmp, pixelsMap);

    var probability = new Dictionary<byte, double>();
    foreach (KeyValuePair<byte, long> entry in pixelsMap)
    {
        probability.Add(entry.Key, entry.Value / (double)numOfPixels);
    }

    var entropy = FindEntropy(probability.Values.ToArray());
    return entropy;
}

3 references
public static void GroupPixels(Bitmap bmp, Dictionary<byte, long> groups)
{
    Rectangle rect = new Rectangle(0, 0, bmp.Width, bmp.Height);
    BitmapData bmpData = bmp.LockBits(rect, ImageLockMode.ReadWrite, bmp.PixelFormat);

    IntPtr ptr = bmpData.Scan0;

    int bytes = Math.Abs(bmpData.Stride) * bmp.Height;
    byte[] rgbValues = new byte[bytes];

    System.Runtime.InteropServices.Marshal.Copy(ptr, rgbValues, 0, bytes);

    for (int counter = 0; counter < rgbValues.Length; counter += 3)
    {
        var r = rgbValues[counter];
        if (groups.TryGetValue(r, out var numberOfPixels))
        {
            groups[r]++;
        }
        else
        {
            groups.Add(r, 1);
        }
    }

    System.Runtime.InteropServices.Marshal.Copy(rgbValues, 0, ptr, bytes);

    bmp.UnlockBits(bmpData);
}
```

2) Зображення, що використовувались для тестування, отримані результати та їх порівняльна оцінка

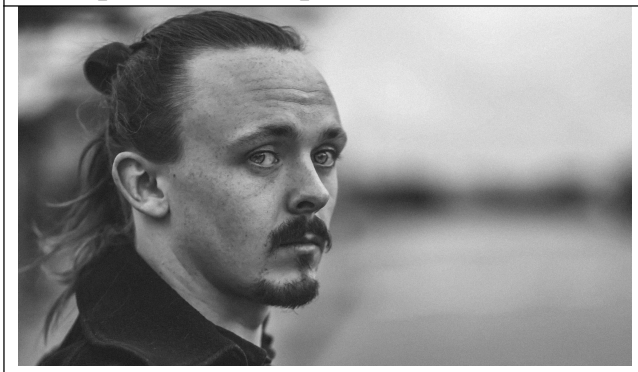
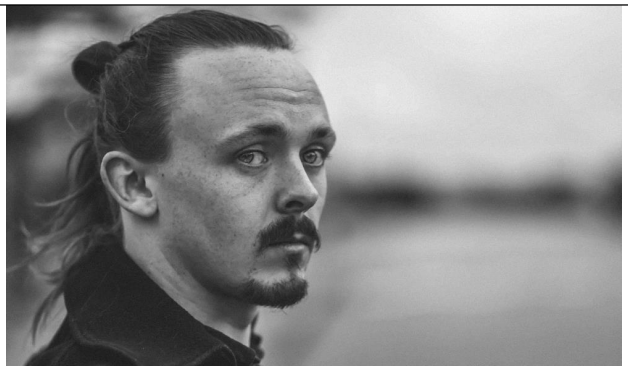
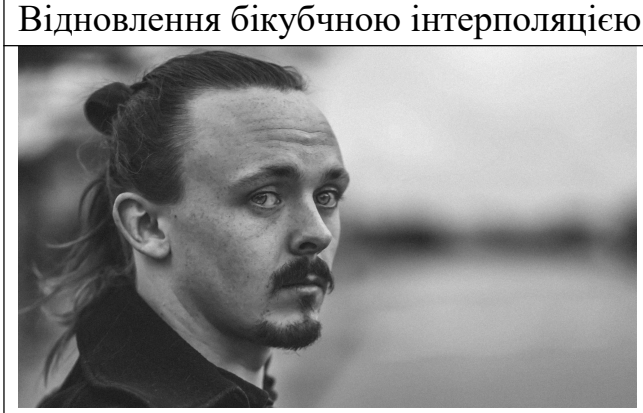
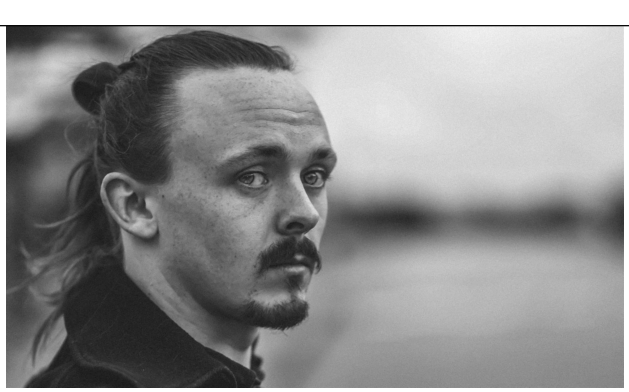
Початкове зображення

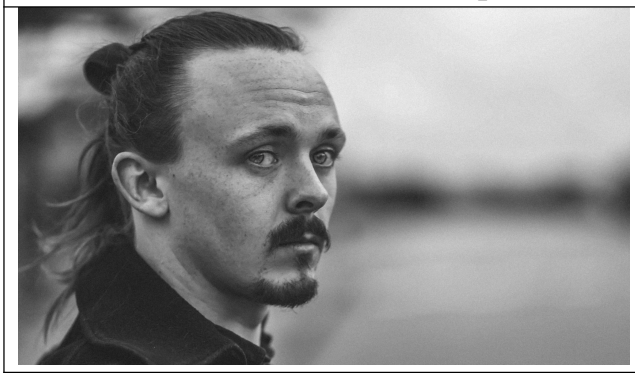
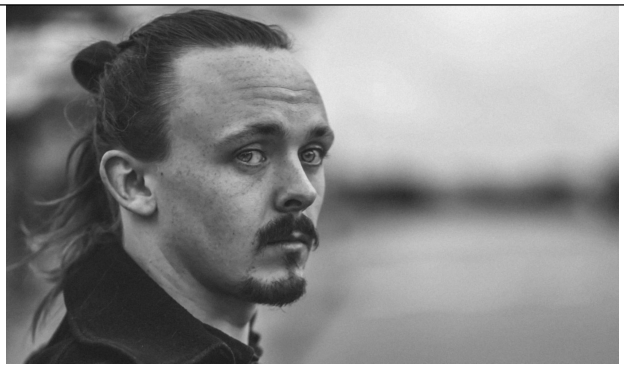


Розширення: 3995X2336

Розширення було зменшено в 2 і 4 рази

Також це фото було квантованим на 8, 16, 64 рівні а потім відновлений білінійною і бікубічною інтерполяцією.

Дискретизація з кроком 2	Дискретизація з кроком 4
	
Відновлення бікубчною інтерполяцією	
	

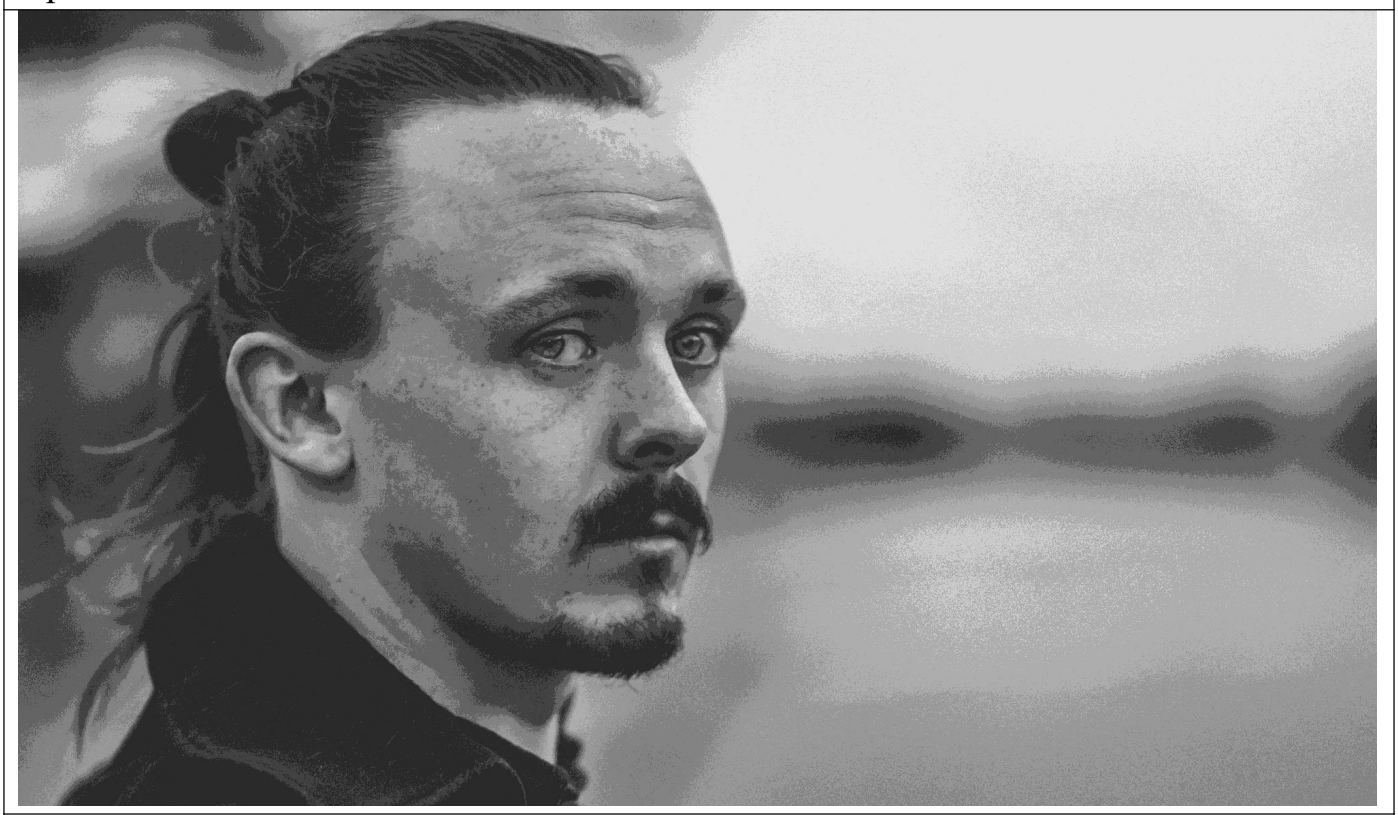
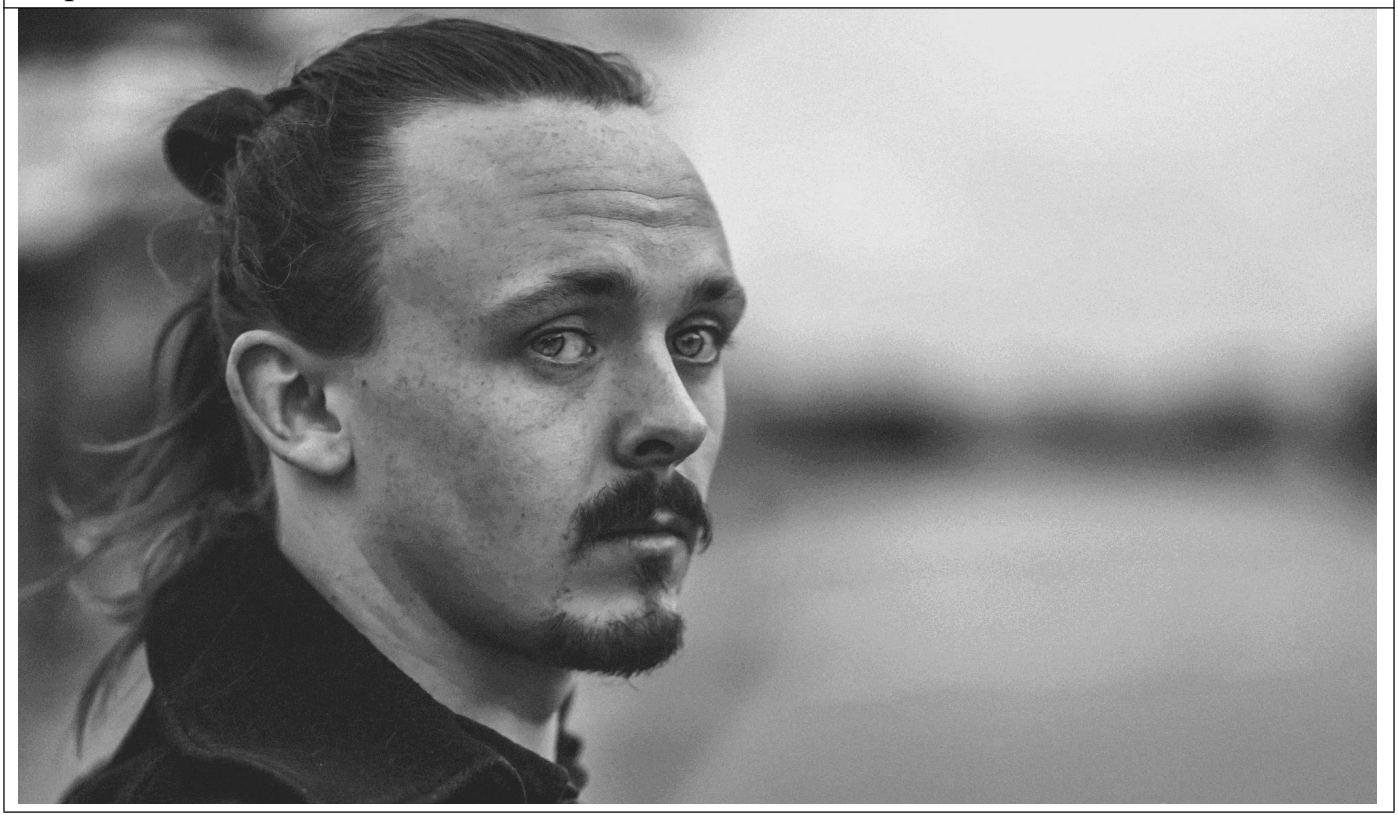
Відновлення білінійною інтерполяцією	
	

```
Microsoft Visual Studio Debug Console

Entropy of image: 7,72653042506422
Entropy of sampled 2 image: 8,192272511314862
Entropy of sampled 4 image: 8,14220806374289
```

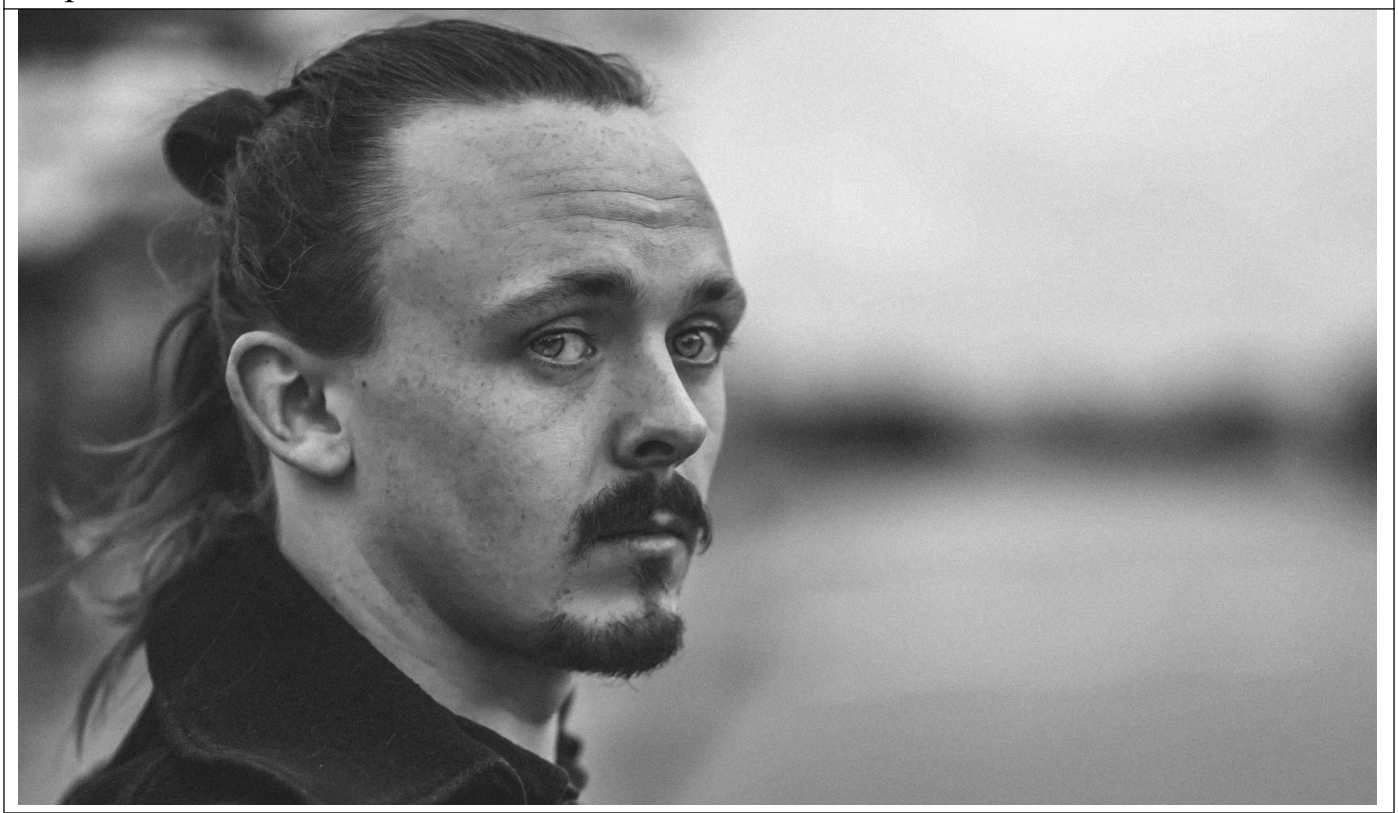
```
--------------------------------------------------------------------------------------------------------

Entropy of restored_2_bicubic: 8,178612512985291
Relative entropy: ∞

Entropy of restored_2_bilinear: 8,168737248200024
Relative entropy: ∞


--------------------------------------------------------------------------------------------------------

Entropy of restored_4_bicubic: 8,137458333041867
Relative entropy: ∞

Entropy of restored_4_bilinear: 8,125266534839794
Relative entropy: ∞
```

Згідно отриманих даних можна визначити що при дискретизації у даному випадку втратилась частина інформації яка не відновилась у процесі інтерполяції, саме тому відносна ентропія є безкінечною, також можна визначити те що бікубічна інтерполяція продокує більший об’єм інформації. Також варто відзначити що при дискретизації не зменшилось різноманіття інформації (тобто не зменшиось число рівнів), це також видно з ентропії. Чому ж ентропія зображення на відновлених фотографіях нтща ніж на зжатих? Це очікувана поведінка оскільки число пікселів зросло зарахунок “домальованих” які брались з уже відомих отже число пікселів зросло, число унікальних ні, тому ентропія зменшилась. Також з даних можна зробити висновок що бікубічна інтерполяція трішки деталізованіше обробляє зображення.

Квантування зображення

8 рівнів

16 рівнів


64 рівні



```
----------------------------------------------------------------------------------------------------

Entropy of quntized_8: 1,066392383528072
Relative entropy: ∞


----------------------------------------------------------------------------------------------------

Entropy of quntized_16: 1,3225240286666904
Relative entropy: ∞


----------------------------------------------------------------------------------------------------

Entropy of quntized_64: 2,526924610972623
Relative entropy: ∞


----------------------------------------------------------------------------------------------------
```

Виходячи з представлених вище даних можна зазначити що квантування дуже суттєво впливає на ентропію, і це не дивно оскільки кількість унікальних пікселів зменшується і кількість рівнів також і порогів також, що дуже суттєво впливає на фото. Також тут теж є дані які втратились тому відносна ентропія дорівнює безкінечності.

Висновки

Отже тут було перевірено два способи обробки зображень, дискретизація та квантування. Судячи з даних які я отримав для зменшення етропії зображень краще використовувати квантування. Також при обробці обома способами було повністю втрачено інформацію про пікселі з деякою інтенсивністю.

Github: https://github.com/325Vitalik/TI