

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА
ПОЛІТЕХНІКА”**

Кафедра систем штучного інтелекту

Лабораторна робота № 1

з дисципліни «Теорія Інформації»

Варіант 31 (1)

Виконав:

студент групи КН-213

Ярмусь Віталій

Викладач:

Косаревич Р. Я.

Львів – 2020 р.

Тема: ЕКСПЕРИМЕНТАЛЬНЕ ВИЗНАЧЕННЯ ЕНТРОПІЇ ПОВІДОМЛЕННЯ

Мета роботи: Вивчення властивостей ентропії як кількісної міри інформації.

Хід роботи

1. Знайти ентропію дискретної випадкової величини X , заданої розподілом $P(x_i)$. Значення $P(x_i)$ взяти з таблиці 2.1 згідно варіанту.


i	0,15	0,07	0,09	0,08	0,09	0,12	0,04	0,13	0,12	0,10
---	------	------	------	------	------	------	------	------	------	------

Ентропія обчислюється за формулою:

$$-\sum_{i=1}^K p(x_i) \log(p(x_i))$$

```
- references
static void Main(string[] args)
{
    var px = new double[] { 0.15, 0.07, 0.09, 0.08, 0.09, 0.12, 0.04, 0.13, 0.12, 0.10 };
    Console.WriteLine("Entropy of sequence: " + FindEntropy(px) + "\n\n");
}
```

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

 Microsoft Visual Studio Debug Console

Entropy of sequence: 3,230641720711895

2. Розробити програму для визначення ентропії зображення. В програмі передбачити наступні функції:
- а. читання файлу зображення;

```
Bitmap bmp = new Bitmap("C:\\Users\\Vitalii\\OneDrive\\University\\3 семестр\\Теорія Інформації\\1\\assets\\harrison-qi-PBFtL7_RFJk-unsplash.jpg");
```



- б. Обрахунок загального висла пікселів

```
var numOfPixels = bmp.Width * bmp.Height;  
Console.WriteLine("Number of pixels: " + numOfPixels);
```

```
Number of pixels: 20250000
```

с. Перетворення зображення в чорно-біле

В даному випадку на значення червоного, синього, та зеленого каналу виставляється одне значення що дозволяє створити чорно-біле зображення.

```
bmp = ChangePixels(bmp);
```

```
1 reference
public static Bitmap ChangePixels(Bitmap bmp)
{
    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];
        var g = rgbValues[counter + 1];
        var b = rgbValues[counter + 2];
        var tmp = (byte)(r * 0.3 + g * 0.59 + b * 0.11);

        rgbValues[counter] = tmp;
        rgbValues[counter + 1] = tmp;
        rgbValues[counter + 2] = tmp;
    }

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

    bmp.UnlockBits(bmpData);

    return bmp;
}
```

d. Збереження зображень у різних форматах

```
bmp.Save("C:\\Users\\Vitalii\\OneDrive\\University\\3 семестр\\Теорія Інформації\\1\\assets\\grey.jpg", ImageFormat.Jpeg);
bmp.Save("C:\\Users\\Vitalii\\OneDrive\\University\\3 семестр\\Теорія Інформації\\1\\assets\\grey.png", ImageFormat.Png);
bmp.Save("C:\\Users\\Vitalii\\OneDrive\\University\\3 семестр\\Теорія Інформації\\1\\assets\\grey.tiff", ImageFormat.Tiff);
bmp.Save("C:\\Users\\Vitalii\\OneDrive\\University\\3 семестр\\Теорія Інформації\\1\\assets\\grey.bmp", ImageFormat.Bmp);
Console.WriteLine("Images saved");
```



✓ grey.bmp



✓ grey.jpg



✓ grey.png



✓ grey.tiff

- е. Групування пікселів для кожного формату за інтенсивністю (тут береться значення зеленого але це не має значення оскільки у всіх каналах однакове значення див. 2.с)

```
var imageTypes = new string[] { "jpg", "png", "tiff", "bmp" };

foreach (var type in imageTypes)
{
    Console.WriteLine($"\\nTYPE: {type}");
    bmp = new Bitmap($"C:\\Users\\Vitalii\\OneDrive\\University\\3 семестр\\Теорія Інформації\\1\\assets\\grey.{type}");

    var pixelsMap = new Dictionary<byte, long>();
    GroupPixels(bmp, pixelsMap);
}
```

```
1 reference
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);
}
```

- ф. Розрахунок імовірностей значень інтенсивностей свічіння груп пікселів

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

g. Загальна сума імовіностей повинна дорівнювати 1

```
var sumOfValues = probability.Values.Sum();  
Console.WriteLine("Sum of probabilities: " + sumOfValues);
```

h. Знаходження ентропії

```
var entropy = FindEntropy(probability.Values.ToArray());  
Console.WriteLine("Entropy: " + entropy);
```

```
Number of pixels: 20250000  
Images saved  
  
TYPE: jpg  
Sum of probabilities: 1,0002962962962967  
Entropy: 7,285890706339597  
  
TYPE: png  
Sum of probabilities: 1,000296296296297  
Entropy: 7,273437456894616  
E  
TYPE: tiff  
Sum of probabilities: 1,000296296296297  
Entropy: 7,273437456894616  
  
TYPE: bmp  
Sum of probabilities: 1,000296296296297  
Entropy: 7,273437456894616
```

На мові C# нема ні встроєного методу для обчислення ентропії, ні нормальних пакетів тому для перевірки результатів було обраховано ентропію з допомогою мови python і пакету scikit-image





```
import skimage  
import skimage.measure  
from skimage import io  
  
I = io.imread('assets\grey.bmp')  
print('bmp: ' + str(skimage.measure.shannon_entropy(I)))  
  
I = io.imread('assets\grey.jpg')  
print('jpg: ' + str(skimage.measure.shannon_entropy(I)))  
  
I = io.imread('assets\grey.tiff')  
print('tiff: ' + str(skimage.measure.shannon_entropy(I)))  
  
I = io.imread('assets\grey.png')  
print('png: ' + str(skimage.measure.shannon_entropy(I)))
```

```
bmp: 7.2699646652540695  
jpg: 7.2824203021002045  
tiff: 7.2699646652540695  
png: 7.2699646652540695
```

Результати практично ідентичні. Похибка не більша ніж 0.003

3. Висновки

Як видно з результатів обрахунку ентропії типи bmp tiff png зберігають однакову кількість інформації, це тому що вони використовують стиснення без втрат (tiff, png) або не використовують стиснення (bmp). Jpeg в свою чергу використовує стиснення і тому ми бачимо інший результат. Також варто звернути увагу на об'єм пам'яті який займають ці файли.

 grey.bmp	✓	11.10.2020 17:23	BMP File	59 344 KB
 grey.jpg	✓	10.10.2020 20:10	JPG File	2 751 KB
 grey.png	✓	10.10.2020 20:15	PNG File	22 142 KB
 grey.tiff	✓	10.10.2020 20:15	TIFF File	21 050 KB

З фото вище видно що формати які використовують стиснення без втрат займають в 10+ разів більше пам'яті.

Ентропія напряму залежить від імовірності виникнення події, чим більш подія передбачувана тим менше корисних даних вона несе, тим меншою буде ентропія, а з цього випливає що чим більша ентропія тим більше інформації несе повідомлення і тим ваще його запам'ятати

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