Міністерство освіти і науки України

НТУУ «Київський політехнічний інститут ім. Ігоря Сікорського»

Фізико-технічний інститут

Лабораторна робота №1

З дисципліни

«Автоматизація обробки ІзОД»

Варіант 10

**Виконав:**

Студент 5 курсу ФТІ

групи ФЕ-91мп

Пугач І. О.

**Перевірив:**

Прогонов Д. О.

Київ-2020

**І. Підготовка**

**Вхідні дані**

Тестовий пакет – MIRFlickr-20k (https://press.liacs.nl/mirflickr/#sec\_download)

Вибірка зображень – 250 зображень;

Формування вибірки зображень – псевдовипадкове, з використанням генератора Мерсена (стартове значення співпадає з номером студента в загальному списку групи) за модулем кількості зображень в тестовому пакеті.

**Завдання**

1. Сформувати тестову вибірку зображень з вихідного пакета;

2. Для кожного каналу кольору кожного зображення з тестового пакета обчислити наступні характеристики:

a. Максимальна / мінімальне значення;

b. Математичне сподівання і дисперсію;

c. Медіану значень, інтерквартільний розмах;

d. Коефіцієнти асиметрії та ексцесу (нормалізований);

3. Для кожного каналу кольору кожного зображення з тестового пакета побудувати гістограму значень яскравості пікселів;

4. Провести апроксимацію отриманих гістограм з використанням відомих імовірнісних розподілів, визначити найкращу апроксимацію;

5. Побудувати розподіл типів використаних імовірнісних розподілів для яких досягається мінімальне значення помилки апроксимації з п.4.

**ІI. Хід роботи**

Роботу виконуватимемо мовою Python за допомогою блокового інтерпретатора Jupyter. Також в роботі будуть використані такі бібліотеки як:

* Os
* Matplotlib
* Numpy
* Scipy
* Pandas
* Sklearn

1. **Формування тестової вибірки зображень з вихідного пакета**

Для цього скористаємося функцією numpy.random.choices() що обирає випадкові числа з переданого масива за допомогою генератора Мерсена. Також задамо початкове значення варіату за допомогою функції numpy.random.RandomState()

from os import listdir

from matplotlib import image

import random

import numpy as np

# load all images in a directory

np.random.RandomState(10)

random\_indexes = np.random.choice(range(25000), 250)

loaded\_images = list()

for i in range(250):

# load image

filename = 'im' + str(random\_indexes[i]) + '.jpg'

img\_data = image.imread('C:/Users/Parzival/Desktop/mirflickr/' + filename)

# store loaded image

loaded\_images.append(img\_data)

print('> loaded %s %s' % (filename, img\_data.shape))

Після цього ориманий масив зображень буде знаходитись в loaded\_images в виді двомірного масиву з трьома значеннями яскравості в кожній комірці.

Тепер сформуємо матрицю для збору статистичних даних. Для цього створимо двомірний numpy масив на три рядки для кожного каналу кольору та на 256 стовпчиків, що відповідатиме кількості пікселів відповідної яскравості.

#1.

values = np.zeros((3, 256))

index = 0

for image in loaded\_images:

for i in range(image.shape[0]):

for j in range(image.shape[1]):

values[0][image[i][j][0]] += 1

values[1][image[i][j][1]] += 1

values[2][image[i][j][2]] += 1

index += 1

if (index % 10) == 0:

print('> processed %i images' % (index))

1. **Знаходження статистичних даних**
2. **Максимальна / мінімальне значення**

**RED = 0**

**GREEN = 1**

**BLUE = 2**

**#2.a. max and min finding**

**print('Red:\t', end='')**

**for i in range(255, 0, -1):**

**if (values[RED][i] != 0):**

**print('Max - {}, '.format(i), end='')**

**break**

**for i in range(0, 255):**

**if (values[RED][i] != 0):**

**print('Min - {}'.format(i))**

**break**

**print('Green:\t', end='')**

**for i in range(255, 0, -1):**

**if (values[GREEN][i] != 0):**

**print('Max - {}, '.format(i), end='')**

**break**

**for i in range(0, 255):**

**if (values[GREEN][i] != 0):**

**print('Min - {}'.format(i))**

**break**

**print('Blue:\t', end='')**

**for i in range(255, 0, -1):**

**if (values[BLUE][i] != 0):**

**print('Max - {}, '.format(i), end='')**

**break**

**for i in range(0, 255):**

**if (values[BLUE][i] != 0):**

**print('Min - {}'.format(i))**

**break**

Маючи вихідний масив з кількістю пікселів відповідної яскравості для знаходженння максимального значення потрібно йти з кінця масиву до першого ненульового значення. Його індекс і казатиме про наявність піксклів відповідної яскравості. Для мінімального потрібно проробити те саме але з початку. Після виконання коду отимаємо наступні значення:

**Red: Max - 255, Min - 0**

**Green: Max - 255, Min - 0**

**Blue: Max - 255, Min – 0**

1. **Математичне сподівання і дисперсія**

Для знаходження скористаємось відповідними формулами:

- Математичне очікування



- Дисперсія

Де xi наше значення яскравості, а pi – ймовірність її появи. pi можна знайти як кількість пікселів даної яскравості поділену на всю кількуість пікселів

# Mat ochikuvannya i dispersiya

#RED

sum\_val = sum(values[RED])

M\_red = 0

for index in range(len(values[RED])):

p = (values[RED][index] / sum\_val)

M\_red += p \* index

D\_red = 0

for index in range(len(values[RED])):

p = (values[RED][index] / sum\_val)

D\_red += p \* ((index - M\_red) \*\* 2)

print("Red: mat ochikuvannya - {0:.2f}, dispersiya - {1:.2f}"\

.format(M\_red, D\_red))

#GREEN

sum\_val = sum(values[GREEN])

M\_green = 0

for index in range(len(values[GREEN])):

p = (values[GREEN][index] / sum\_val)

M\_green += p \* index

D\_green = 0

for index in range(len(values[GREEN])):

p = (values[GREEN][index] / sum\_val)

D\_green += p \* ((index - M\_green) \*\* 2)

print("Green: mat ochikuvannya - {0:.2f}, dispersiya - {1:.2f}"\

.format(M\_green, D\_green))

#BLUE

sum\_val = sum(values[BLUE])

M\_blue = 0

for index in range(len(values[BLUE])):

p = (values[BLUE][index] / sum\_val)

M\_blue += p \* index

D\_blue = 0

for index in range(len(values[BLUE])):

p = (values[BLUE][index] / sum\_val)

D\_blue += p \* ((index - M\_blue) \*\* 2)

print("Blue: mat ochikuvannya - {0:.2f}, dispersiya - {1:.2f}"\

.format(M\_blue, D\_blue))

Отримуємо:

**Red: mat ochikuvannya - 114.31, dispersiya - 5889.79**

**Green: mat ochikuvannya - 104.24, dispersiya - 5549.14**

**Blue: mat ochikuvannya - 93.26, dispersiya - 5960.26**

**с. Медіана значень та інтерквартальний розмах.**

Для медіани бідемо сумувати всі кількості пікселів в масиві і коли сума перевалить за половину кількості пікселів – ми будемо знати де знаходиться медіана.

def medianFinder(array\_values):

sum\_val = sum(array\_values)

summa = 0

for index in range(len(array\_values)):

summa += array\_values[index]

if (summa > (sum\_val / 2)):

return index - 1

if (summa == (sum\_val / 2)):

return (2 \* index + 1) / 2

Для інтерквартального розмаху скористаємось тією ж логікою але будемо шукати вже сумарну імовірність пройдених пікселів. В передане значення передаємо 0.25 для пошуку Q1 та 0.75 для пошуку Q3. Їх різниця і буде інтерквартальним розмахом.

def quartFinder(array\_values, quart):

sum\_val = sum(array\_values)

summa = 0

for index in range(len(array\_values)):

prev\_sum = summa

p = array\_values[index] / sum\_val

summa += p

if (summa > quart):

if (quart - prev\_sum < summa - quart):

return index - 1

elif (quart - prev\_sum > summa - quart):

return index

else:

return (2 \* index - 1) / 2

Маємо:

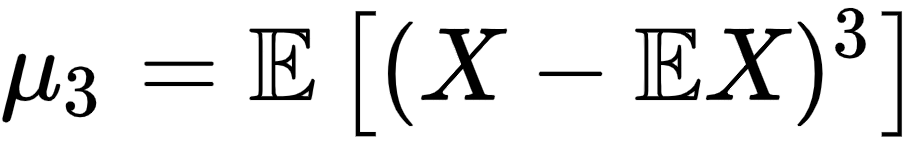
**Red: median - 111, IQR - 132**

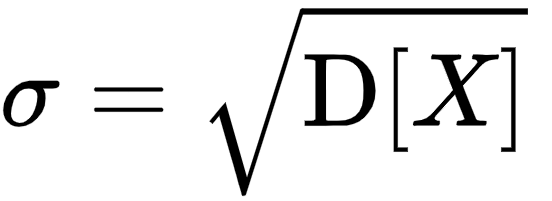
**Green: median - 96, IQR - 127**

**Blue: median - 74, IQR - 135**

**d. Коефіцієнти асиметрії та ексцесу**

Для цього також використаємо відповідні формули коефіцієнту асиметрії:





Для спрощення напишемо загальну формулу для оператора Е, що приймає степінь та мат очікування.

def E\_operator(arr\_values, M, power):

sum\_val = sum(arr\_values)

ans = 0

for index in range(len(arr\_values)):

p = (arr\_values[index] / sum\_val)

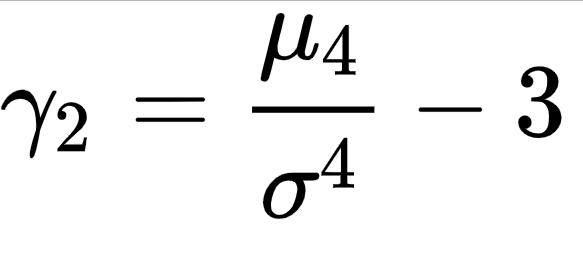
ans += p \* ((index - M) \*\* power)

return ans

Тепер формула отримає вигляд:

Asym\_red = E\_operator(values[RED], M\_red, 3) / (D\_red \*\* (3 / 2))

Та формули коефіцієнту ексцесу:



Ekscess\_red = E\_operator(values[RED], M\_red, 4) / (D\_red \*\* 2) - 3

Отримуємо:

**Red: Asimmetriya - 0.128, Ekscess - 1.836**

**Green: Asimmetriya - 0.292, Ekscess - 1.932**

**Blue: Asimmetriya - 0.495, Ekscess - 1.978**

1. **Для кожного каналу кольору кожного зображення з тестового пакета побудувати гістограму значень яскравості пікселів**

Для цього скористаємося бібліотекою matplotlib:

import matplotlib.pyplot as plt

x = range(256)

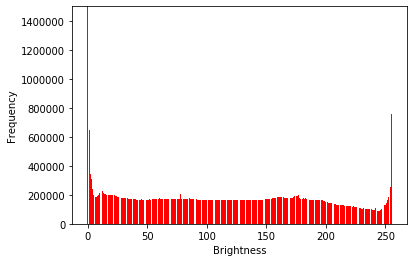
plt.bar(x,values[RED],color='red')

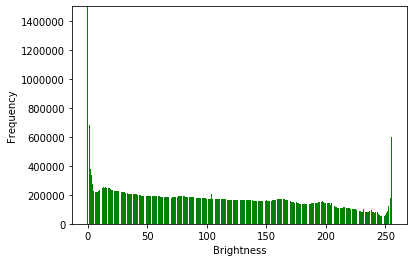
plt.xlabel('Brightness')

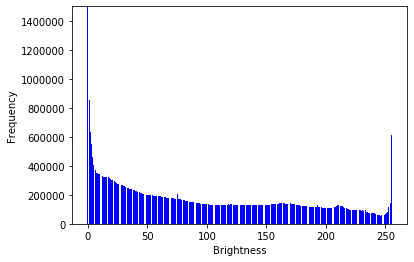
plt.ylabel('Frequency')

plt.ylim(top=1500000)

plt.show()







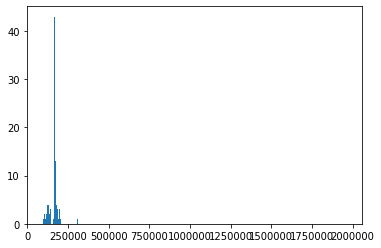
1. **Провести апроксимацію отриманих гістограм з використанням відомих імовірнісних розподілів, визначити найкращу апроксимацію**

**import seaborn as sns**

**test = values[RED]**

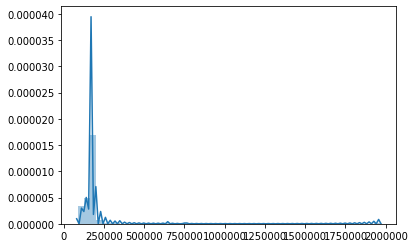
**plt.hist(test,bins = 1000)**

**plt.show()**

****

Для цього скористаємося можливостями бібліотеки seaborn

X,Y = sns.distplot(np.ravel(test)).get\_lines()[0].get\_data()



Побудована апроксимація описує наші дані, тепер спробуємо використати відомі розподіли для опису нашого

plt.plot(lnspc, pdf\_g, label="Norm",color = 'grey')

plt.plot(lnspc, pdf\_gamma, label="Gamma",color = 'blue')

plt.plot(lnspc, pdf\_beta, label="Beta",color = 'red')



Як бачимо найближче до нашого розподілу апроксимує Gamma.

**ІІІ. Висновки**

В даній лабораторній роботі було проаналізовано вибірку з 250 зображень датасету MIRFlickr-20k. Було знайдено що всі канали охоплюють увесь спектр значень. Було знайдено мат. очікування – 114 для чевоного каналу, 104 для зеленого і 93 для синьогоканалу відповідно і дисперсію 5889.79, 5549.14 та 5960.26. Медіану значень та інтерквартальний розподіл.

Також коефіцієнти асиметрії було знайдено, та вони були додатніми, що каже про нахил графіка в бік спадання значень. Також коефіцієнт ексцесу був від’ємний для всіх розподілів.

Графіки приведено за допомогою бібліотеки matplotlib, вони повністю відповідають знайденим значенням.

При знаходженні апроксимації було показано топ відомих розподілів по схожості з нашим. Було отримано що gamma-розподіл підходить краще інших.