МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. І.СІКОРСЬКОГО»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра фізико-технічних засобів захисту інформації

Лабораторна робота № 2

з дисципліни: «Автоматизація обробки ІзОД»

Варіант №3

Керівник: Виконав:

Прогонов Дмитро Олександрович студент 5 курсу групи ФЕ-91мп

Захищено з оцінкою Євдокимов О. О.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, підпис

Київ – 2020 р.***Вихідні 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.

***Лабораторна робота №2***

1. Сформувати тестову вибірку зображень з вихідного пакета;
2. Для кожного каналу кольору кожного зображення з тестового пакета обчислити наступні характеристики:
   1. Математичне сподівання і дисперсію;
   2. Коефіцієнти асиметрії та ексцесу (нормалізований);
3. Побудувати вектори параметрів зображень, що складаються з:
   1. Математичних очікувань значень яскравості для кожного каналу кольору;
   2. Математичних очікувань і дисперсії значень яскравості для кожного каналу кольору;
   3. Математичних очікувань, дисперсії і коефіцієнта асиметрії значень яскравості для кожного каналу кольору;
   4. Математичних очікувань, дисперсії, коефіцієнтів асиметрії та ексцесу значень яскравості для кожного каналу кольору;
4. Побудувати гаусові моделі зображень з використанням розрахованих раніше параметрів.
5. Провести декомпозицію кожного каналу кольору кожного зображення з застосуванням методу головних компонент (PCA):
   1. Варіюючи кількість компонент, провести реконструкцію окремих каналів кольору зображень (від компонент з найбільшою енергією поступово переходячи до компонентів з мінімальною енергією).
   2. Побудувати залежність помилки відновлення (середнє відхилення вихідного зображення відреконструйованого, MSE) від кількості використаних компонент.
6. Провести моделювання окремих каналів кольору зображень з використанням марковських ланцюгів:
   1. Для кожного каналу кольору кожного зображення розрахувати стохастическую матрицю марковської ланцюга першого і другого порядків (обробка пікселів по горизонталі справа наліво і навпаки, а також по вертикалі зверху вниз і навпаки). У звіті привести явний вигляд однієї марковської ланцюга для одного з каналів кольору тестового зображення;

Перевірити властивість регулярності, реккурентное і незворотності (irreducible) для отриманих марковских моделей для 5 ітерацій.

**Хід роботи**

**1. Формування тестової вибірки зображеннь**

Лабораторна робота буде виконана мовою Python за допомогою блокового інтерпритатора Jupyter. В данній лабораторній роботі використовувались наступні бібліотеки:

* Scipy
* Matplotlib
* Matplotlib.pyplot
* Scipy.ndimage
* Random
* Os
* Numpy
* Pandas
* PIL
* Networkx

Зробимо вибірку псевдовипадкової кількості 250 кольорових зображень із пакету mirflick та сформуємо матрицю для збору статистичних даних про зобьраження. Для цього створимо двомірний Numpy масив на три рядки для кожного каналу кольору та на 256 стовпчиків, що відповідатиме кількості пікселів відповідної яскравості

*# 1.generate a test sample of images from the source package*

np.random.RandomState(3)

sample\_size = 250

indexes = np.random.choice(range(25000), sample\_size)

images = list()

**for** i **in** range(sample\_size):

filename = 'im' + str(indexes[i]) + '.jpg'

img\_data = img.imread('.../downloads/mirflickr/' + filename)

images.append(img\_data)

print(f'**{sample\_size}** images uploaded, processing begins...')

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

**for** image **in** 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

print('all images processed')

**2. Обчислення характеристик для кожного каналу зображень**

1. Математичне сподівання і дисперсія

Розрахунки будуть проводитись за наступними формулами:



(1),



(2),

Де (1) - математичне очікування, а (2) – дисперсія, xi - значення яскравості, pi – ймовірність її появи. pi можна знайти як кількість пікселів даної яскравості поділену на всю кількість пікселів

expectations = {

'RED': 0,

'GREEN': 0,

'BLUE': 0

}

variances = {

'RED': 0,

'GREEN': 0,

'BLUE': 0

}

**for** color **in** colors:

print(color + ':')

sum\_values = sum(values[colors[str(color)]])

expectation = 0

**for** i **in** range(len(values[colors[str(color)]])):

p = (values[colors[str(color)]][i] / sum\_values)

expectation += p \* i

variance = 0

**for** i **in** range(len(values[colors[str(color)]])):

p = (values[colors[str(color)]][i] / sum\_values)

variance += p \* ((i - expectation) \*\* 2)

expectations[str(color)] = expectation

variances[str(color)] = variance

print(f'expectation - **{expectation}**')

print(f'variance - **{variance}**')

Таким чином отримуэмо значення Мат. очікування та дисперсії:

RED:

expectation - 118.0212951406293

variance - 6046.004677937074

GREEN:

expectation - 110.16358232767844

variance - 5723.162584113702

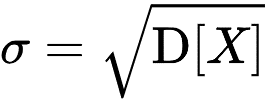
BLUE:

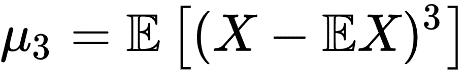
expectation - 100.94022637322685

variance - 6117.5026395624645

1. Коефіцієнти асиметрії та ексцесу

Використаємо відповідні формули коефіцієнту асиметрії:





Реалізуємо даний функціонал за допомогою такого коду:

**def** finder(array, expectation, power):

summa\_values = sum(array)

final = 0

**for** i **in** range(len(array)):

p = (array[i] / summa\_values)

final += p \* ((i - expectation) \*\* power)

**return** final

**for** color **in** colors:

print(color + ':')

asymmetry = finder(values[colors[color]], expectations[color], 3) / (variances[color] \*\* (3 / 2))

excess = finder(values[colors[color]], expectations[color], 4) / (variances[color] \*\* 2) - 3

print(f'asymmetry - **{asymmetry}**')

print(f'excess - **{excess}**')

Отримаємо:

RED:

asymmetry - 0.10142663138630191

excess - -1.2024401239735634

GREEN:

asymmetry - 0.21888462250400514

excess - -1.1213572009721986

BLUE:

asymmetry - 0.39093833763692565

excess - -1.0861327638833809

* + 1. **Побудувати вектори параметрів зображень:**

Сформуємо масив з векторами всіх потрібних нам значень, та використаемо окремі частини для розрахунку:

vector = np.array([np.array([expectations['RED'], variances['RED'], asymmetries['RED'], excesses['RED']]),

np.array([expectations['GREEN'], variances['GREEN'], asymmetries['GREEN'], excesses['GREEN']]),

np.array([expectations['BLUE'], variances['BLUE'], asymmetries['BLUE'], excesses['BLUE']])])

vector = np.copy(vector)

**for** im **in** images:

im = np.reshape(im, (-1, 3))

im = np.swapaxes(im, 0, 1)

vector = np.concatenate((vector,im),axis=1)

* + 1. **Побудувати гаусові моделі зображень з використанням розрахованих раніше параметрів.**

Сформуємо гаусові моделі. Отримаємо вектори параметрів зображень, що складаються з:

1. Математичних очікувань значень яскравості для кожного каналу кольору;

Виведемо інформацію за допомогою коду:

x1 = np.random.normal(expectations['GREEN'], variances['GREEN'], 1)

print("expectations of brightness values:**\n**" + str(x1))

Отримали:

expectations of brightness values:

[-2747.03159443]

1. Математичних очікувань і дисперсії значень яскравості для кожного каналу кольору;

Виведемо інформацію за допомогою коду:

x2 = np.cov(vector)

print("expectations and variance of brightness values:**\n**" + str(x2[:2, :2]))

expectations and variance of brightness values:

[[6046.77719226 5220.911815 ] [5220.911815 5723.85514798]]

1. Математичних очікувань, дисперсії і коефіцієнта асиметрії значень яскравості для кожного каналу кольору;

expectations and variance of brightness values:

[[6046.77719226 5220.911815 4645.47605222]

[5220.911815 5723.85514798 5379.53147581]

[4645.47605222 5379.53147581 6118.29823466]]

1. Математичних очікувань, дисперсії, коефіцієнтів асиметрії та ексцесу значень яскравості для кожного каналу кольору;

expectations, variance, asymmetry and excess brightness values:

[[6046.77719226 5220.911815 4645.47605222]

[5220.911815 5723.85514798 5379.53147581]

[4645.47605222 5379.53147581 6118.29823466]]

* + 1. **Провести декомпозицію кожного каналу кольору кожного зображення з застосуванням методу головних компонент (PCA)**

Даний процес проходть для визначеної кількості компонентів n, яке ми також передаватимемо як параметр.

**def** PCA(img\_2d, nmpc):

cov\_mat = img\_2d - np.mean(img\_2d)

val, vec = np.linalg.eigh(np.cov(cov\_mat))

p = np.size(vec, axis =1)

i = np.argsort(val)

i = i[::-1]

vec = vec[:, i]

val = val[i]

**if** (nmpc <p) **or** (nmpc >0):

vec = vec[:, range(nmpc)]

score = np.dot(vec.T, cov\_mat)

rcn = np.dot(vec, score) + np.mean(img\_2d).T

rcn\_img\_mat = np.uint8(np.absolute(rcn))

**return** rcn\_img\_mat

number\_of\_comp = [5, 30, 80]

fig=plt.figure(figsize=(20,20))

fig.add\_subplot(1, 2, 1)

plt.title('original')

plt.imshow(images[3])

k = 2

**for** number **in** number\_of\_comp:

test\_r\_rcn, test\_g\_rcn, test\_b\_rcn = PCA(test\_r, number),

PCA(test\_g, number), PCA(test\_b, number)

rcn\_img = np.dstack((test\_r\_rcn, test\_g\_rcn, test\_b\_rcn))

fig.add\_subplot(3, 2, k)

plt.title(f'**{number}** components')

plt.imshow(rcn\_img)

k += 2

Отримаємо наступний результат:



Рисунок 1 **–** відновлені фото з різною кількістю компонентів

Опрацювавши результати можна зробити висновок, що зі збільшенням кількості компонентів росте якість відновлення зображення.

Далі виконаємо функцію для багатьох кроків та поріняємо початкове фото з відновленим за допомогою функції середньої квадратичної похибки.

**def** my\_mse(first\_img, second\_img):

err = np.sum((first\_img.astype("float") - second\_img.astype("float")) \*\* 2)

err /= float(first\_img.shape[0] \* first\_img.shape[1])

**return** err

all\_mse = list()

**for** i **in** range(100):

test\_r\_recon, test\_g\_recon, test\_b\_recon = PCA(test\_r, i), PCA(test\_g, i), PCA(test\_b, i)

rcn\_img = np.dstack((test\_r\_recon, test\_g\_recon, test\_b\_recon))

all\_mse.append(my\_mse(test, rcn\_img))

plt.plot(range(len(all\_mse)),all\_mse)

plt.xlabel("components")

plt.ylabel("MSE")

plt.show()

В результаті отримуеємо графік:

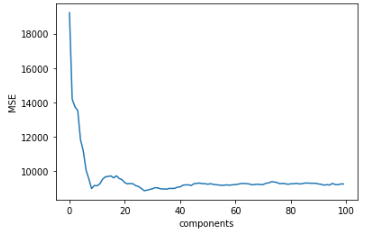


Рисунок 2 – залежність MSE відновлених фото від кількості компонент

З даного графіка видно що залежність має експоненціальний характер та похибка дуже значно зменшується при збільшенні компонент.

* + 1. **Провести моделювання окремих каналів кольору зображень з використанням марковських ланцюгів**

1. Для кожного каналу кольору кожного зображення розрахувати стохастичну матрицю марковського ланцюга першого і другого порядків;

Для побудови марківського ланцюга сформуємо матрицю 256 на 256 та пройшови по всіх пікселях картинки запишемо кількість переходів між ними. При цьому представимо два типи обходу матриць “C-type”:

matrix1 = np.zeros(shape=(256, 256))

array = test\_r.flatten()

prev\_color = arr[0]

**for** i **in** range(len(array) - 1):

matrix1[array[i]][array[i + 1]] += 1

matrix = matrix1[0] / sum(matrix1[0])

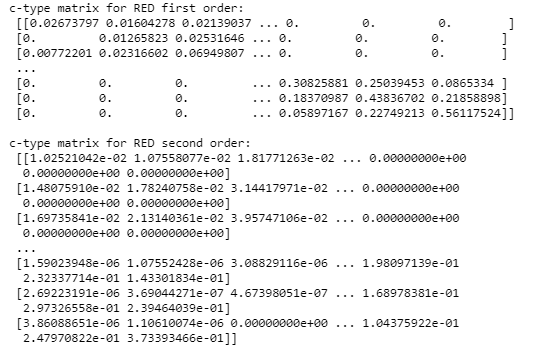
**for** i **in** range(1, 256):

matrix = np.vstack((matrix, matrix1[i] / sum(matrix1[i])))

print("c-type matrix for RED first order:**\n**", matrix)

print("**\n**c-type matrix for RED second order:**\n**", np.linalg.matrix\_power(matrix, 2))

Отримаємо наступні матриці:



Та Fortran-type:

test\_list = [test\_r, test\_g, test\_b]

col = ['RED', 'GREEN', 'BLUE']

c = 0

**for** color **in** test\_list:

matrix1 = np.zeros(shape=(256, 256))

*#Fortran-type*

array = color.flatten('F')

prev\_color = array[0]

**for** i **in** range(len(array) - 1):

matrix1[array[i]][array[i + 1]] += 1

matrix = matrix1[0] / sum(matrix1[0])

**for** i **in** range(1, 256):

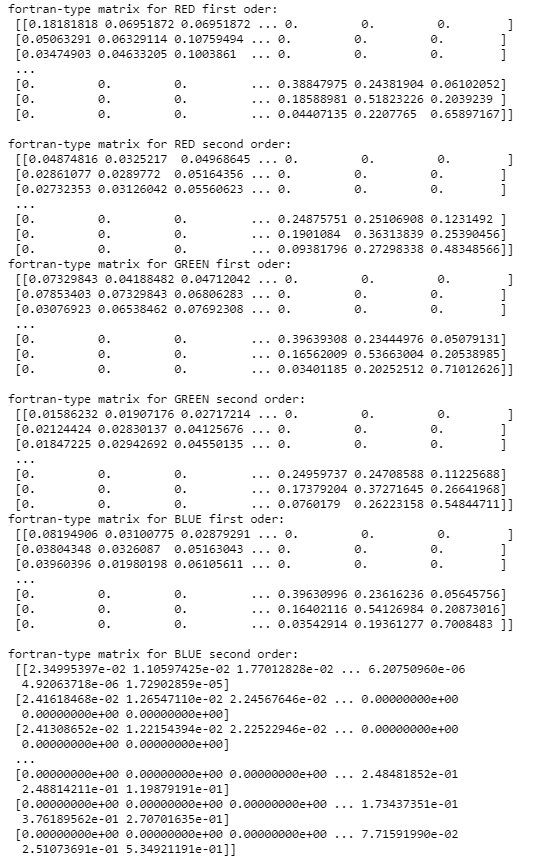
matrix = np.vstack((matrix, matrix1[i] / sum(matrix1[i])))

print(f'fortran-type matrix for **{col[c]}** first oder:**\n**', matrix)

print(f'**\n**fortran-type matrix for **{col[c]}** second order:**\n**', np.linalg.matrix\_power(matrix, 2))

c +=1

Отримаємо наступні матриці:



Побудуємо графічне представлення ланцюга. Пікселі будуть вузлами, а переходи між ними – з’єднаннями. Та Fortran-type

data = np.triu(matrix) + np.triu(matrix).T index = [str(i) **for** i **in** range(data.shape[0])] dataframe = pd.DataFrame(data, index=index, columns=index) plt.figure(1,figsize=(12,12)) g = nx.from\_pandas\_adjacency(dataframe) nx.draw(g, with\_labels=**True**) plt.show()

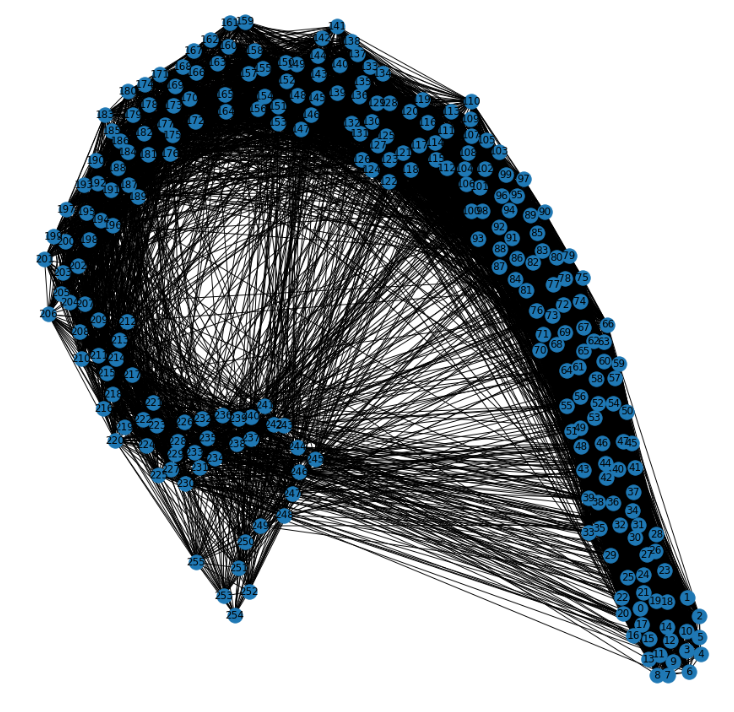


Рисунок 3 - Вигляд марківського ланцюга

**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 – 118.6 для чевоного каналу, 110.3 для зеленого і 100.59 для синьогоканалу відповідно і дисперсію 6046.877, 5723.17 та 6117.50. Також знайдено коефіцієнти асиметрії та ексцесу:

Червоний: асиметрія - 0.10, ексцес -1.20

Зелений: асиметрія - 0.218, ексцес -1.12

Синій: асиметрія - 0.39, ексцес – 1.08

Побудувано вектори параметрів зображень та знайдено Гаусовські моделі для одновимірного та багатовимірних варіантів в залежності від кількості даних.

За допомогою методу головних компонент було відновлено тестові зображення та показано, що при збільшенні кількості компонент зростає якість відновлення (рис. 1).

Зібравши дані, було побудовано графік залежності середньої квадратичної похибки відновлених дображень від кількості компонентів (рис. 2).

Було помічено експоненціальну залежність, що свідчить про значні зміни при невеликих кількостях компонентів (< 20) та майже непомітні при великих значеннях (> 100).

Проведуно моделювання окремих каналів кольору зображень з використанням марковських ланцюгів, та сформовано стохастичні матриці за різними типами обходів. За даними було побуловано графічну можель марківського ланцюга.

З (рис. 3) видно скупчення схожих яскравостей та плавний перехід від великих значень до малих. Це говорить про відсутність різких зміщень в кольоровій гамі пікселів зображень.