#### 

### **Перед початком виконання роботи:**

* Рекомендовано використовувати Python та Jupyter Notebook для 2 завдання
* В описі до кожного завдання вказується набір рекомендованих бібліотек та корисні посилання
* Оцінювання:

1. Завдання 1 **0.5 б**.
2. Завдання 2 **4 б**.
3. Завдання 3 **1.5 б**.
4. Захист лабораторної роботи **4 б.**
5. Теоретичні питання **2 б.**

#### **Завдання 1: Обчислення власних значень та власних векторів матриці (0.5 б.).**

#### **Опис:** Напишіть функцію, яка приймає квадратну матрицю і повертає її власні значення та власні вектори, використовуючи бібліотеку NumPy. Також здійсніть перевірку рівності A⋅v=λ⋅v для кожного власного значення та відповідного власного вектора. [Відео](https://www.youtube.com/watch?v=PFDu9oVAE-g&t=806s), яке допоможе уявити процес та згадати необхідну теорію**.**

### **Завдання 2. Реалізація методу зменшення розмірності зображення за допомогою аналізу головних компонент (PCA: Image Compression) (4 б.).**

**Опис:** Реалізувати функцію для зменшення розмірності зображення за допомогою аналізу головних компонент (PCA).

Необхідні бібліотеки для виконання завдання: NumPy, matplotlib, Scikit-learn або їх аналоги.

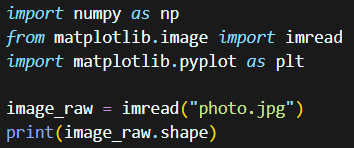
Корисні посилання: [коротке введення до PCA](https://www.youtube.com/watch?v=HMOI_lkzW08), [для тих хто хоче глибше зрозуміти суть PCA))](https://www.youtube.com/watch?v=5v4CozbY1_0), [посилання на обговорення у форумі на цю тему](https://www.quora.com/How-does-principal-component-analysis-work-in-image-processing)

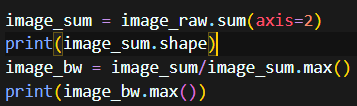
**Що таке PCA?**

Аналіз основних компонентів (PCA) — це техніка, яка використовується в обробці зображень для зменшення його розмірності. У PCA зображення представляється як матриця піксельних значень, і алгоритм ідентифікує головні компоненти зображення шляхом знаходження власних векторів коваріаційної матриці піксельних значень. Ці власні вектори представляють найважливіші характеристики або шаблони в зображенні. Зменшуючи розмірність даних зображення за допомогою PCA, стає легше стискати та зберігати зображення, а також виконувати такі операції, як розпізнавання та класифікація зображень.

**Алгоритм виконання завдання:**

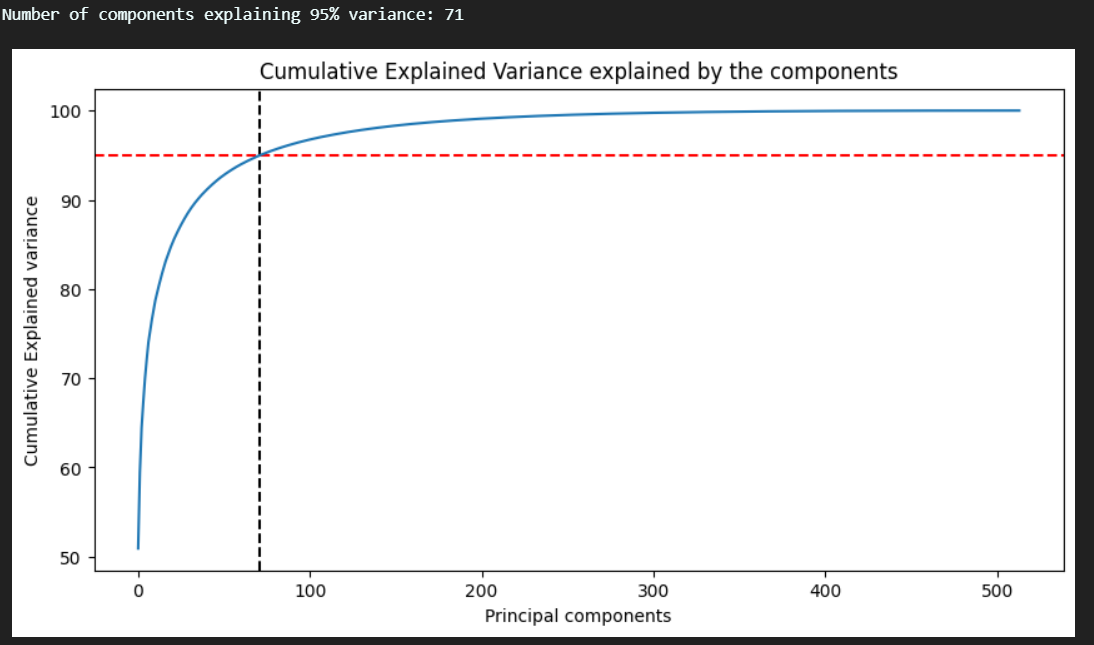
1. Вивести початкове кольорове зображення та вектор, що буде містити: розміри зображення в пікселях та кількість основних каналів кольорів, що використовуються

Приклад знаходження цього вектора: 

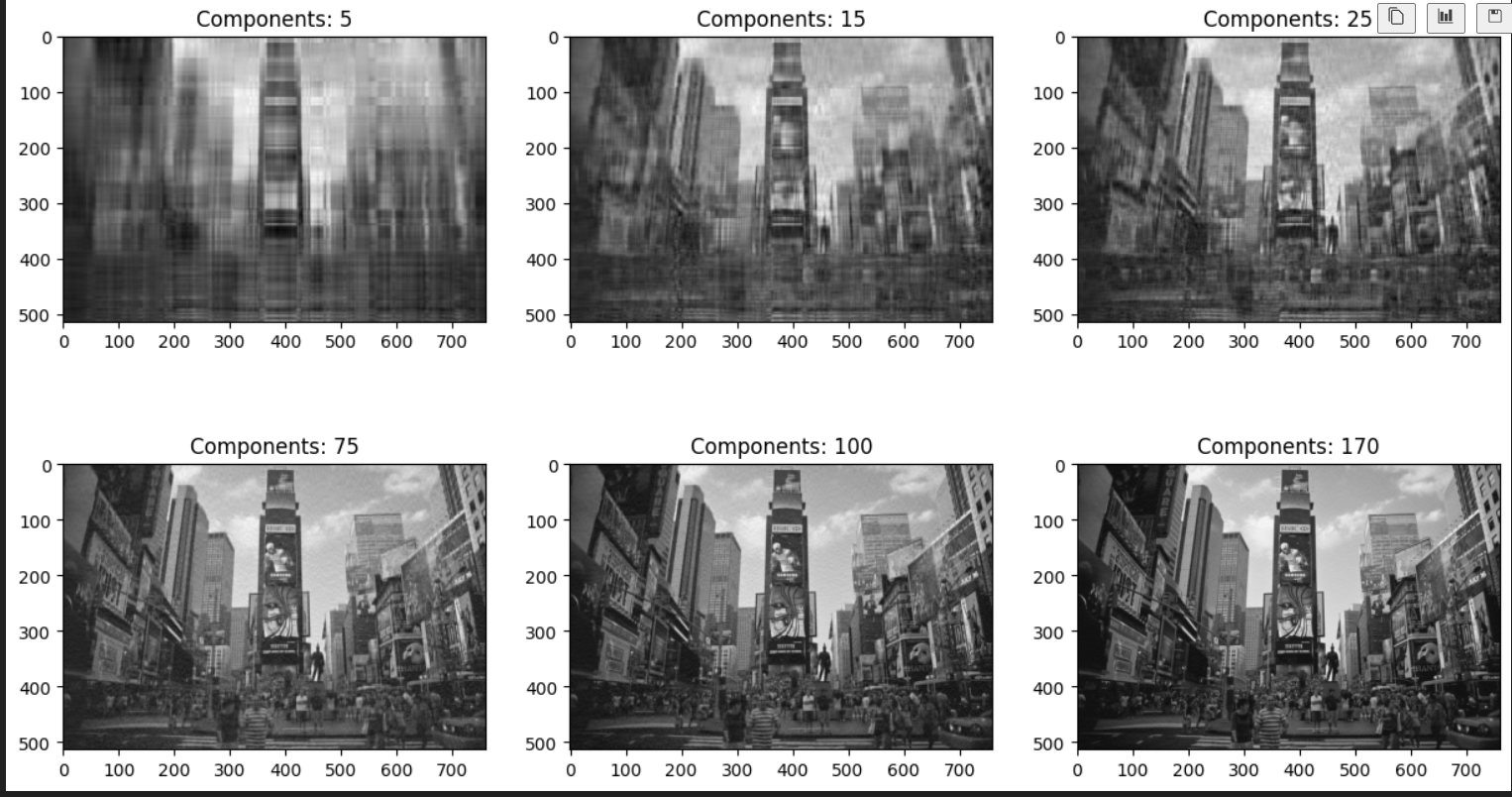
1. Перетворити зображення в чорно-біле та вивести розмір зображення і кількість каналів кольорів **(0.25 б.)**  
   



1. Застосувати PCA для матриці компонентів image\_bw. Вивести cumulative variance та знайти кількість компонент, які необхідні для покриття 95% of the variance. Використовуйте бібліотечні засоби (рекомендовано NumPy) **(0.75 б.).**

Вивести графік відповідного процесу: **додаткові 0.5 б.  
**

Провести реконструкцію чорно-білого зображення, використовуючи обмежену кількість компонентів, знайдену в попередньому кроці. Вивести отримане зображення. Для 95% покриття даних очікувано отримати більш чітке зображення, чи не так? Зауважте, що ми точно зафіксували всі основні елементи – ви все ще можете дуже добре ідентифікувати об’єкти. Чого не вистачає, так це чіткості — та, можливо, саме дрібні деталі у візуальних елементах роблять зображення привабливим і чітким. **1б.**  


1. Проведіть реконструкцію зображення для різної кількості компонент та виведіть відповідні результати. Спробуйте взяти більшу кількість компонент та виведіть відповідний результат. Чи отримали ви більш чітке зображення? А якщо взяти меншу кількість компонент? **(2 б).**
2. 

#### **Завдання 3: Використання діагоналізації, власних значень та векторів в криптографії. Опис:** Використайте діагоналізацію для розшифрування кодів**.** Рекомендується використовувати бібліотеку Numpy. Корисні посилання: [Короткий опис як можна застосовувати принципи лінійної алгебри в криптографії](https://www.youtube.com/watch?v=S_2MV3ncHj0),[відео про саму криптографію](https://www.youtube.com/watch?v=-yFZGF8FHSg) **1.5 б. Завдання:**

#### Створити функцію decrypt\_message(encrypted\_vector, key\_matrix), яка розшифровує зашифрований вектор encrypted\_vector за допомогою матриці ключа key\_matrix, використовуючи обернену операцію діагоналізації.

#### Функція кодування задається таким чином:

#### 

#### Перевірити роботу розроблених функцій на прикладі випадково згенерованої матриці ключа та текстового повідомлення.

#### **Приклад виконання:** Original Message: Hello, World! Encrypt Message: [118703.+1.04957181e-11j 180926.+1.50897913e-11j 149312.+1.90968912e-11j

#### 161873.+1.52982299e-11j 188078.+1.70487245e-12j 145036.+1.64536004e-11j

#### 139370.+4.15241913e-11j 195037.+1.51003306e-11j 155629.+1.38144125e-11j

#### 206859.+1.38439864e-11j 145588.+1.95891393e-11j 130163.+1.55598216e-11j

#### 176423.+1.26512259e-11j] Decrypted Message: Hello, World! **Теоретичні питання 2б.:**

#### Що таке власне значення і власний вектор матриці? Як вони обчислюються? **0.25 б.**

#### Які властивості мають власні вектори симетричних матриць? **0.25 б.**

#### Які можуть бути недоліки використання PCA, і які стратегії можуть використовуватися для подолання цих недоліків? **0.75 б**.

#### Які переваги має діагоналізація матриці в криптографії? Як вона застосовується для шифрування та дешифрування повідомлень? **0.75 б.**

#### 

#### 