SPRAWOZDANIE

Zajęcia: Matematyka konkretna

Prowadzący: prof. dr hab. Vasyl Martsenyuk

|  |  |
| --- | --- |
| Laboratorium Nr 1  Data 11.10.2025 Temat: „Analiza macierzowa. Podstawowe poje˛cia. Rozk lad SVD” Wariant 1 | Marika Daniszewska  Informatyka  II stopień, stacjonarne,  2 semestr, gr. 1a |

1. Polecenie: wariant 1 zadania

W ramach laboratorium należało wykonać zadanie dotyczące kompresji obrazu metodą SVD (Singular Value Decomposition).  
Celem było zbadanie, w jaki sposób rozkład SVD może być wykorzystany do redukcji danych obrazowych przy zachowaniu możliwie wysokiej jakości obrazu.

W szczególności należało:

* wczytać dowolny obraz i przekształcić go do skali szarości,
* przeprowadzić rozkład SVD dla macierzy obrazu,
* obliczyć, ile wartości singularnych należy zachować, aby odtworzony obraz zachował 90% energii (informacji),
* porównać wyniki rekonstrukcji obrazu dla różnych wartości k (liczby zachowanych wartości singularnych),
* obliczyć przybliżony współczynnik kompresji oraz przeanalizować wpływ kompresji na jakość obrazu.

2. Opis programu opracowanego (kody źródłowe, zrzuty ekranu)

 Wczytanie obrazu  
Obraz (1.webp) jest wczytywany z pliku i konwertowany do skali szarości.  
Macierz obrazu jest zapisywana jako tablica typu numpy.ndarray.

 Rozkład SVD  
Program wykonuje rozkład:

A=UΣVTA = U \Sigma V^TA=UΣVT

gdzie A to macierz obrazu, U i V – macierze ortogonalne, a Σ – macierz diagonalna zawierająca wartości singularne.

 Analiza energii  
Obliczana jest suma kwadratów wartości singularnych oraz wykres skumulowanej energii.  
Na jego podstawie określane jest minimalne k, które zapewnia zachowanie 90% całkowitej energii obrazu.

 Rekonstrukcja i porównanie  
Program rekonstruuje obraz przy różnych wartościach k (np. 5, 20, 50, 100, k₉₀, 200).  
Każdy zrekonstruowany obraz jest zapisywany do katalogu svd\_results/ i prezentowany na jednym wykresie obok oryginału.

 Obliczenie współczynnika kompresji  
Szacowany współczynnik kompresji określa stosunek liczby pikseli w oryginalnym obrazie do liczby parametrów potrzebnych do zapisu obrazu w formie zredukowanego rozkładu SVD.

# Importy i funkcje pomocnicze

import numpy as np

import matplotlib.pyplot as plt

from PIL import Image

import os

from pathlib import Path

# Funkcja do wczytania obrazu i konwersji do skali szarości (float)

def load\_image\_gray(path):

    img = Image.open(path).convert('L')  # 'L' -> grayscale

    arr = np.array(img).astype(float)

    return arr, img.size  # size = (width, height)

# Funkcja do zapisu macierzy jako obrazu uint8

def save\_image\_from\_array(arr, out\_path):

    arr\_clipped = np.clip(arr, 0, 255).astype(np.uint8)

    Image.fromarray(arr\_clipped).save(out\_path)

# Funkcja do rekonstrukcji przy użyciu k wartości singularnych

def svd\_reconstruct(A, k):

    U, S, Vt = np.linalg.svd(A, full\_matrices=False)

    Uk = U[:, :k]

    Sk = np.diag(S[:k])

    Vtk = Vt[:k, :]

    return Uk @ Sk @ Vtk

# Funkcja do obliczenia minimalnego k by zachować energy\_ratio (np. 0.9)

def find\_k\_for\_energy(S, energy\_ratio=0.9):

    singular\_vals\_sq = S\*\*2

    total\_energy = singular\_vals\_sq.sum()

    cumulative = np.cumsum(singular\_vals\_sq)

    k = np.searchsorted(cumulative, energy\_ratio \* total\_energy) + 1

    return k, total\_energy, cumulative

# Ścieżka do obrazu

image\_path = '1.webp'

out\_dir = 'wyniki'

os.makedirs(out\_dir, exist\_ok=True)

A, size = load\_image\_gray(image\_path)

h, w = A.shape

print(f"Rozmiar obrazu (h, w): {A.shape}, rozmiar z Pillow (width, height): {size}")

# Pokaż oryginalny (grayscale)

plt.figure(figsize=(6,6))

plt.imshow(A, cmap='gray')

plt.title('Oryginalny obraz (skala szarości)')

plt.axis('off')

plt.show()

# Obliczanie SVD (full\_matrices=False dla wydajności)

U, S, Vt = np.linalg.svd(A, full\_matrices=False)

print('Liczba singularnych wartości:', S.shape[0])

# Znajdź k dla 90% energii

energy\_target = 0.90

k\_90, total\_energy, cumulative = find\_k\_for\_energy(S, energy\_ratio=energy\_target)

print(f"Minimalne k potrzebne by zachować {energy\_target\*100:.0f}% energii: {k\_90} z {S.shape[0]}")

# Wykres skumulowanej energii

plt.figure(figsize=(6,4))

plt.plot(np.arange(1, len(S)+1), np.cumsum(S\*\*2) / total\_energy, marker='o', markersize=3)

plt.xlabel('k (liczba singularnych wartości)')

plt.ylabel('Skumulowana energia (udział całkowitej energii)')

plt.grid(True)

plt.axvline(k\_90, color='r', linestyle='--', label=f'k\_90 = {k\_90}')

plt.legend()

plt.title('Skumulowana energia według liczby singularnych wartości')

plt.show()

# Wybrane wartości k do demonstracji: małe, k\_90, większe (ograniczone do długości S)

k\_values = [5, 20, k\_90, 50, 100, min(len(S), 200)]

k\_values = sorted(list(set([min(int(k), len(S)) for k in k\_values])))

recons = {}

for k in k\_values:

    rec = svd\_reconstruct(A, k)

    recons[k] = rec

    out\_path = os.path.join(out\_dir, f'recon\_k\_{k}.png')

    save\_image\_from\_array(rec, out\_path)

    print(f"Zapisano rekonstrukcję dla k={k} -> {out\_path} (rozmiar: {rec.shape})")

# Pokaż porównanie (ograniczone do max 6 kolumn)

cols = len(recons) + 1

plt.figure(figsize=(3\*cols, 4))

plt.subplot(1, cols, 1)

plt.imshow(A, cmap='gray')

plt.title('Oryginał')

plt.axis('off')

i = 2

for k, rec in sorted(recons.items()):

    plt.subplot(1, cols, i)

    plt.imshow(rec, cmap='gray')

    plt.title(f'k={k}')

    plt.axis('off')

    i += 1

plt.tight\_layout()

plt.show()

# Obliczanie współczynnika kompresji dla k (przybliżonego)

def compression\_ratio(h, w, k):

    original = h \* w

    compressed = h\*k + k + k\*w

    return original / compressed

cr\_k90 = compression\_ratio(h, w, k\_90)

print(f'Przybliżony współczynnik kompresji dla k={k\_90}: {cr\_k90:.3f} (oryginalne piksele / parametry SVD)')

# Zapisz macierz rekonstrukcji dla k\_90 (już powinna być zapisana)

recon\_k90\_path = os.path.join(out\_dir, f'recon\_k\_{k\_90}.png')

print('Ścieżka do zapisanego pliku rekonstrukcji k\_90:', recon\_k90\_path)

# Lista wygenerowanych plików w katalogu wyników

import glob

files = glob.glob(os.path.join(out\_dir, '\*'))

for f in files:

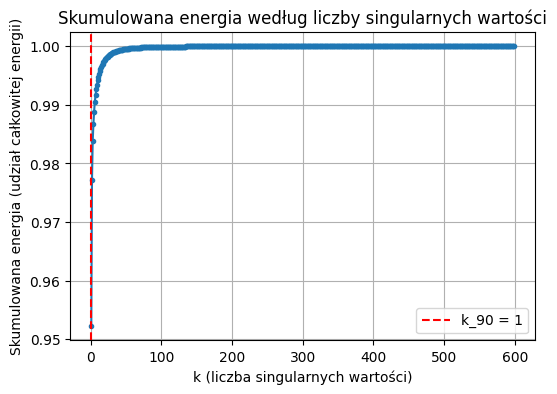
    print('-', f)

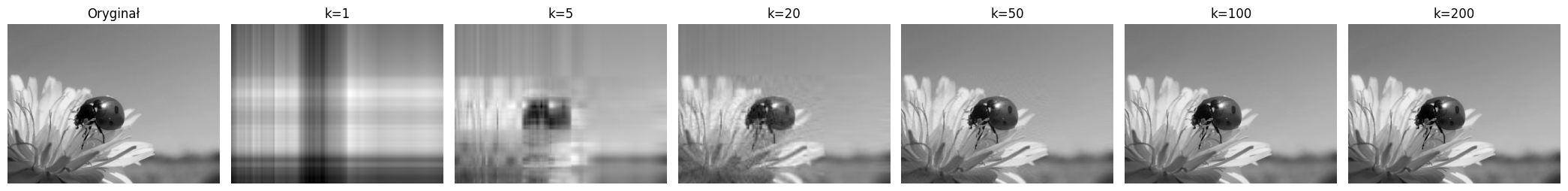
Rozmiar obrazu (h, w): (599, 800), rozmiar z Pillow (width, height): (800, 599)



Liczba singularnych wartości: 599

Minimalne k potrzebne by zachować 90% energii: 1 z 599





Przybliżony współczynnik kompresji dla k=1: 342.286 (oryginalne piksele / parametry SVD)

Github: <https://github.com/MarikaDan/MK/tree/main/LAB1>

3. Wnioski

 Metoda SVD pozwala w efektywny sposób kompresować obrazy, zachowując dużą część informacji wizualnej przy znacznej redukcji danych.

 Jakość obrazu po rekonstrukcji zależy bezpośrednio od liczby zachowanych wartości singularnych k.

* Dla małego k obraz jest mocno rozmazany i traci szczegóły.
* Dla większego k obraz staje się coraz bardziej podobny do oryginału.

 Wartości singularne o największej wartości zawierają najwięcej informacji o strukturze obrazu, a kolejne odpowiadają za szczegóły i szum.

 Zachowanie około 90% energii (informacji) często pozwala uzyskać wizualnie bardzo dobrą jakość przy znacznie mniejszym rozmiarze danych.

 Metoda ta może być z powodzeniem stosowana w praktyce do kompresji obrazów, redukcji wymiarowości danych oraz wstępnego przetwarzania w zadaniach uczenia maszynowego.