Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет систем управления и робототехники

# Лабораторная работа № 6
# "Singular Value Decomposition"

по дисциплине Практическая линейная алгебра

Выполнила: студентка гр. **R3238**

**Нечаева А. А.**

Преподаватель: *Перегудин Алексей Алексеевич*

Санкт-Петербург, 2023-2024

# 1 Сжатие изображений.

Одно из самых наглядных применений сингулярного разложения.

## 1.1 Выбор изображения и подготовка.

Для выполнения работы веберем изображение какого-нибудь покемона, например *Ивизавра*. Далее преобразуем *Ивизавра* к оттенкам серого.

*Рис. 1. Исходное изображение.*

*Листинг 1. Код для преобразования изображения к оттенкам серого*

```
from PIL import Image
import matplotlib.image as mpimg
import numpy as np


img = Image.open('red_eye.png')
black_and_white = img.convert('L')
black_and_white.save('bw_red.png')
```

*Рис. 2. Ивизавр в оттенках серого.*

Далее представим изображение в виде матрицы:

*Листинг 2. Получение изображения в виде матрицы*

```
matrix_image = mpimg.imread('bw_red.png')
```

## 1.2 SVD-разложение полученной матрицы

Функция для выполнения SVD-разложения матрицы на языке *Python*:

*Листинг 3. SVD-разложение матрицы*

```
U, s, W = np.linalg.svd(matrix_image)
s = np.diag(s)
```

Дальше, найдем "укороченные"SVD-разложения этой матрицы, оставив только $k$ первых (наибольших) сингулярных чисел и соотвествующих им векторов.

*Листинг 4. Фрагмент кода для нахождения "укороченных"SVD-разложений и отрисовки результата*

```
k = [20000, 4500, 1000, 250, 100, 50, 20, 10, 5, 1]

for i in k:
    cut = np.dot(np.dot(U[:, :i], s[:i, :i]), W[:i, :])

    fig, axs = plt.subplots(1, 2)

    axs[0].imshow(matrix_image).set_cmap('grey')
    axs[1].imshow(cut).set_cmap('grey')
    axs[0].axis('off')
    axs[1].axis('off')
    axs[0].set_title('Original')
    axs[1].set_title('After compression, k=' + str(i))

    plt.show()
```

Результаты выполнения программы представлены ниже.


a)


b)

*Рис. 3. Сжатие: a) $k = 20000$, b) $k = 4500$.*


a)


b)

*Рис. 4. Сжатие: a) $k = 1000$, b) $k = 250$.*

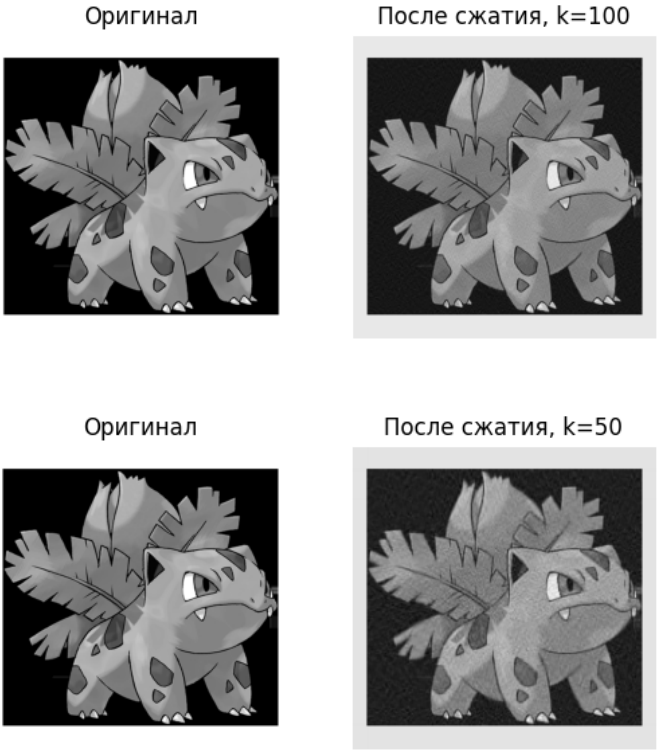


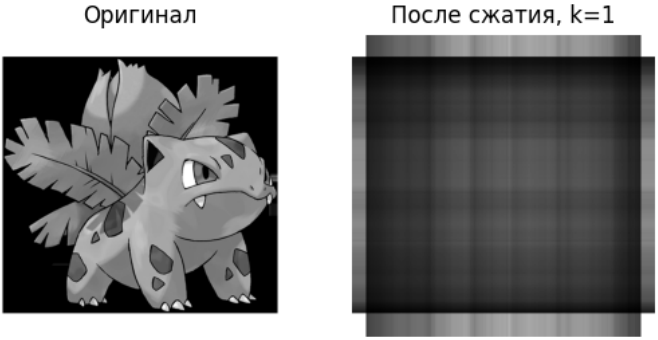

*Рис. 5. Сжатие: a) $k = 100$, b) $k = 50$.*


a)


b)

*Рис. 6. Сжатие: a) $k = 20$, b) $k = 10$.*


a)




b)

*Рис. 7. Сжатие: a) $k = 5$, b) $k = 1$.*

Заметно, что уже при $k \geq 5$ становится возможным различить черты Ивизавра, а при $k \geq 50$ сжатое изображение становится практически неотличимым от оригинала.

Далее проанализируем, сколько памяти мы сэкономили, вычислим "вес"исходного изображения и сжатого. Результаты представлены в таблице 1.

Таблица 1 – Сравнительные результаты работы программы

| k | Различим ли Ивизавр? | Вес оригинала | Вес сжатого | % сжатия |
|---|---|---|---|---|
| $2 \cdot 10^4$ | да, как оригинал | 451725 | 451725 | 0.00 |
| 4500 | да, как оригинал | 451725 | 451725 | 0.00 |
| $10^3$ | да, как оригинал | 451725 | 451725 | 0.00 |
| 250 | да, как оригинал | 451725 | 237750 | 47.37 |
| 100 | да, как оригинал | 451725 | 95100 | 78.95 |
| 50 | да | 451725 | 47550 | 89.47 |
| 20 | да, отдаленно | 451725 | 19020 | 95.79 |
| 10 | да, отдаленно | 451725 | 9510 | 97.89 |
| 5 | нет | 451725 | 4755 | 98.95 |
| 1 | нет, совсем | 451725 | 951 | 99.79 |

## 1.3 Анализ результатов и вывод

Заметим, что при степени сжатия примерно на 90% изображение все еще остается различимым . В частности, при $k \geq 5$ становится возможным различить черты Ивизавра, а при $k \geq 50$ сжатое изображение становится практически неотличимым от оригинала.

Таким образом, можно сделать вывод о том, что применение SVD-разложения является довольно эффективным инструментом сжатия изображений с минимальными потерями качества результата.