Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Застосування SVD алгоритму для розпізнавання облич
(тема)

Виконав:
студент _2_ курсу, групи _ПМм-20-1
Калайчев Г.В.
(прізвище, ініціали)

Спеціальність
113 Прикладна математика
(код і повна назва спеціальності)

Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Прикладна математика
(повна назва освітньої програми)

Керівник ____проф. Сидоров М.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ПМ _____ ____Тевяшев А.Д.
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет _інформаційно-аналітичних технологій та менеджменту
Кафедра _прикладної математики
Рівень вищої освіти _другий (магістерський)
Спеціальність _113 Прикладна математика
(код і повна назва)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _Прикладна математика
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____
(підпис)
“____” _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _Калайчеву Георгію Валентиновичу
(прізвище, ім’я, по батькові)

1. Тема роботи _Застовування SVD алгоритму для розпізнавання облич

затверджена наказом по університету від 05 листопада 2021 р. № 1641 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10 грудня 2021 р.

3. Вихідні дані до роботи _набір даних з фотографіями облич людей

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області
2. Вибір і обґрунтування методу розв’язання
3. Програмна реалізація
4. Результати обчислювального експерименту
5. Аналіз можливих застосувань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Актуальність теми роботи
2. Постановка задачі
3. Аналіз предметної області
4. Метод чисельного аналізу
5. Результати обчислювального експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Підбір та вивчення технічної літератури за темою роботи	8 – 14 листопада 2021 р.	виконано
2	Вибір та обґрунтування методу	15 – 21 листопада 2021 р.	виконано
3	Розробка алгоритму і програми	22 – 28 листопада 2021 р.	виконано
4	Проведення аналітичних досліджень та розрахунків	29 листопада – 5 грудня 2021 р.	виконано
5	Робота над текстом пояснювальної записки	6 – 9 грудня 2021 р.	виконано
6	Представлення роботи на рецензію в ЕК	10 грудня 2021 р.	виконано

Дата видачі завдання _8_ листопада 2021 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Сидоров М.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 50 с., 1 табл., 18 рис., 1 дод., 19 джерел.

МАШИННЕ НАВЧАННЯ, SVD, СИНГУЛЯРНІ ВЕКТОРИ, АЛГОРИТМ МАШИННОГО НАВЧАННЯ, РОЗПІЗНАВАННЯ ОБЛИЧ, ACCURACY, ОРТОГОНАЛЬНІ ВЕКТОРИ, БАЗИСНІ ВЕКТОРИ.

Об’єкт дослідження – набір даних з фотографіями облич людей.

Мета роботи – застосування алгоритму SVD для розв’язання задачі розпізнавання облич.

Методи дослідження – алгоритм розпізнавання облич з використанням SVD.

Кваліфікаційна робота присвячена дослідженню застосування методу розпізнавання облич з використанням алгоритму SVD. Вхідні дані задачі представлені у вигляді набору фотографій облич людей. Розв’язання задачі розпізнавання проходить в три етапи. Перший етап полягає у попередній обробці даних та підготовці їх до подання у алгоритм розпізнавання. Другим етапом є тренування (навчання) моделі на тренувальних даних. Третім етапом є оцінка роботи моделі – отримання результатів прогнозування моделі на тестових даних. Розроблений алгоритм програмно реалізовано за допомогою мови Python у середовищі Jupyter Notebook. Зроблено висновки про результати роботи алгоритму розпізнавання.

ABSTRACT

Introductory note: 50 pages, 1 tables, 18 figures, 1 appendix, 19 sources.

MACHINE LEARNING, SVD, SINGULAR VECTORS, MACHINE LEARNING ALGORITHM, FACE RECOGNITION, ACCURACY, ORTHOGONAL VECTORS, BASES.

The object of research is a set of data with photos of people's faces.

The purpose of the work is to use the SVD algorithm to solve the problem of face recognition.

Research methods – algorithm for face recognition using SVD.

Qualification work is devoted to the study of the application of the method of face recognition using the SVD algorithm. The data for solving the problem is presented in the form of a set of photos of people's faces. The recognition problem is solved in three stages. The first stage is to pre-process the data and prepare them for submission to the recognition algorithm. The second stage is the training (learning) of the model on training data. The third stage is the evaluation of the model – obtaining the results of model prediction on training and test data and comparing them. The developed algorithm is software implemented using the Python language in the Jupyter Notebook environment. Conclusions are made about the results of the recognition algorithm.

ЗМІСТ

ВСТУП

Актуальність теми. Системи біометричної ідентифікації особи [9, 10 , 11] стають все більш необхідними технологіями у наші часи цифровізації. Їх існує досить багато, наприклад, ідентифікація за голосом, відбитками пальців або за зображенням обличчя. Усі вони мають широке застосування у сьогоденні та використовуються у різних сферах життя. Серед усіх біометричних методів ідентифікації ідентифікація за обличчям привернула велику увагу в останні роки, оскільки є ненав'язливою та зручною для користувачів.

За останні роки великий прогрес був досягнутий у розпізнаванні облич [10]. З'явилося дуже багато різноманітних рішень на основі нейронних мереж. Вони є стійкими та точними, можуть працювати у різних умовах освітлення, можуть працювати як з фотографіями, так і з відео. Але урахування усіх цих особливостей потребує дуже багато пам'яті, серйозного та сучасного комп'ютерного обладнання, яке доступне не кожному, що ускладнює їх використання. Більше того, тренування цих моделей нейронних мереж [12] потребує ще більших зусиль починаючи від збирання даних для тренування моделі, закінчуючи обиранням або будуванням архітектури нейронної мережі. Саме тому в цій роботі буде розглянутий алгоритм розпізнавання облич за допомогою SVD [3] (сингулярного розкладу матриці). При використанні цього алгоритму немає необхідності у використанні великих комп'ютерних потужностей та навчанні будь-якої моделі. Це робить цей алгоритм більш швидким у роботі та доступнішим більш широким масам. Отже, актуальним є застосування алгоритму SVD для розв'язання задачі розпізнавання облич.

Мета і завдання кваліфікаційної роботи. Метою кваліфікаційної роботи є розв'язання задачі розпізнавання облич за допомогою алгоритму SVD. Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд і аналіз сучасного стану задачі розпізнавання облич;

– провести огляд основних понять лінійної алгебри, які застосовуються при використанні алгоритму розпізнавання облич на основі SVD;

– застосувати алгоритм розпізнавання облич з використанням SVD;

– скласти алгоритм розв’язання поставленої задачі;

– програмно реалізувати розроблений алгоритм за допомогою мови програмування Python [7] та середовища Jupyter Notebook [8];

– провести низку обчислювальних експериментів для тестових даних.

Об’єктом дослідження є набір даних з фотографіями облич людей.

Предметом дослідження є алгоритм розпізнавання облич з використанням SVD.

Методи дослідження. У кваліфікаційній роботі використовуються методи лінійної алгебри для побудови алгоритму сингулярного розкладу матриці, методи попередньої обробки зображень, алгоритм CascadeClassifier для знаходження обличчя на фотографії, методи оцінки роботи алгоритму розпізнавання облич.

1 АНАЛІЗ МЕТОДІВ РОЗПІЗНАВАННЯ ОБЛИЧ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Огляд методів розв’язання задачі розпізнавання облич

Розпізнавання облич є актуальної задачею у сьогоденні. Саме тому були винайдені багато різних за своїми підходами та особливостями алгоритмів, які здатні розв’язувати поставлену задачу. Вибір алгоритму машинного навчання для розв’язання задачі розпізнавання облич залежить від властивостей та особливостей даних, від складності поставленої задачі. Наприклад, якщо необхідно розпізнавати обличчя на відеозапису, у темному приміщенні, якщо якість зображення низька, тоді необхідно використовувати більш складні алгоритми машинного навчання, які здатні відокремлювати інформацію, яка є на зображенні. Але дуже часто експерти використовують занадто складні за своєю архітектурою, реалізацією та у розгортці на пристрої, на яких буде працювати система розпізнавання облич, нейронні мережі. Це пов’язано з їх популярністю у наші часи та впевненістю у тому, що ці алгоритми спрацюють. Люди позбуваються швидкості у реалізації та зіштовхуються з необхідністю мати більш потужні прилади: комп’ютер, телефон, сервер або будь-який інший прилад, на якому буде тренуватися або використовуватися система розпізнавання облич.

В ході огляду методів розв’язання задачі розпізнавання облич будуть розглянуті наступні підходи до розв’язання цієї задачі:

– алгоритми на основі згорткових нейронних мереж;

– алгоритми з використанням сингулярного розкладу матриці (SVD).

1.1.1 Згорткові нейронні мережі

Згорткові нейронні мережі з’явились в результаті дослідження кори головного мозку. Більш прості архітектури моделей машинного навчання мають

суттєвий недолік – вони не враховують структуру даних, яка може бути добре відома та нести суттєву частину інформації. Зразком такого типу даних може бути зображення. Зображення може бути чорно-білим (тоді це просто масив інтенсивностей) або кольоровим (тоді це масив векторів трьох чисел, що позначають інтенсивність трьох кольорів – червоного, зеленого та синього (RGB) [6]). Подання зображення у вигляді інтенсивностей наведено на рис. 1.1.

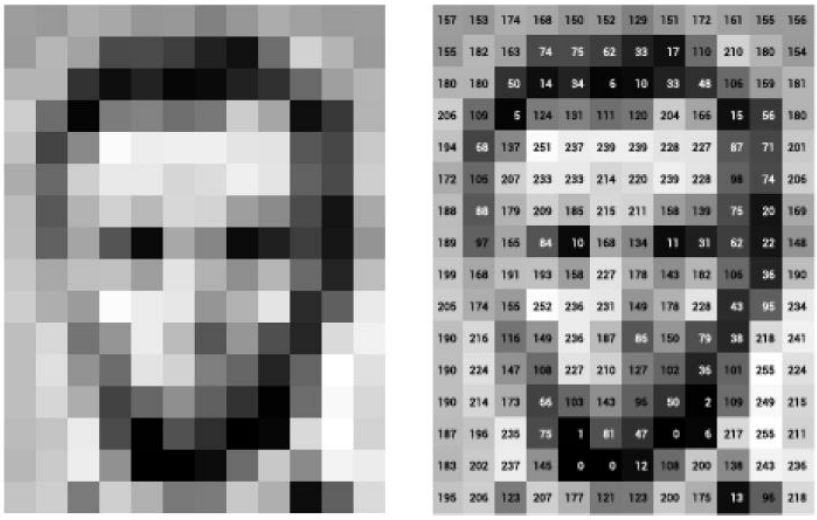

Рисунок 1.1 – Зображення у вигляді інтенсивностей

Згортка [5]– це лінійне перетворення, що застосовується до квадратного вікна та переміщується по вхідним даним, при цьому пікселі вікна скалярно перемножуються на матрицю згортки.

Згортковий прошарок [17] – це основна складова згорткової нейронної мережі. Нейрони на першому згортковому прошарку не зв’язані з кожним пікселем вхідного зображення, а тільки з пікселями у власних рецепторних полях. Але нейрони на другому згортковому прошарку пов’язані з нейронами, що знаходяться в середині невеликого прямокутника в першому прошарку. Сама така архітектура дає можливість нейронній мережі фіксувати та помічати низькорівневі ознаки в першому прихованому прошарку, потім компонувати їх в ознаки більш високого рівня у наступних прихованих прошарках.

Отже, згорткова нейронна мережа приймає на вхід деяке зображення, далі

вона має декілька згорткових прошарків та декілька повно-зв’язних прошарків на кінці для того, щоб зробити деяке передбачення моделі. Приклад загорткової мережі наведено на рис. 1.2.

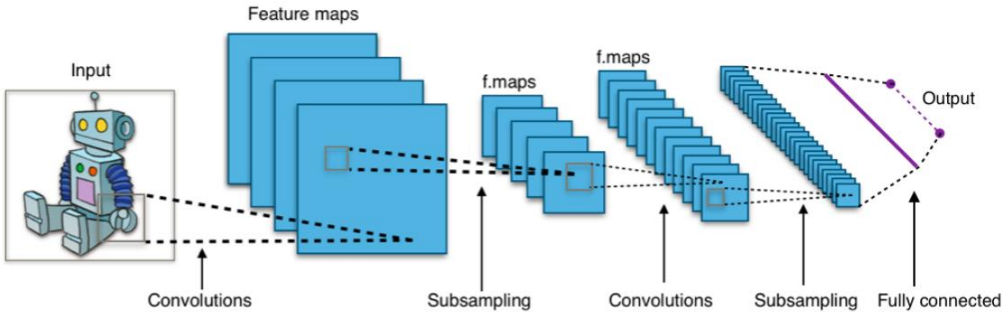

Рисунок 1.2 – Приклад загорткової нейронної мережі

Існує багато готових архітектур на основі згортових нейронних мереж, наприклад, ResNet [11]. Ця нейронна мережа була створена Кеймінгом Хе. Це одна з найпотужніших згорткових нейронних мереж. Вона має велику кількість прихованих прошарків [14] (дуже глибока), а отже, має багато параметрів (мільйони), які треба обробляти. Це робить процес тренування цієї мережі дуже довгим та потребує досить потужного обладнання. Архітектура нейронної мережі ResNet наведена на рис. 1.3.

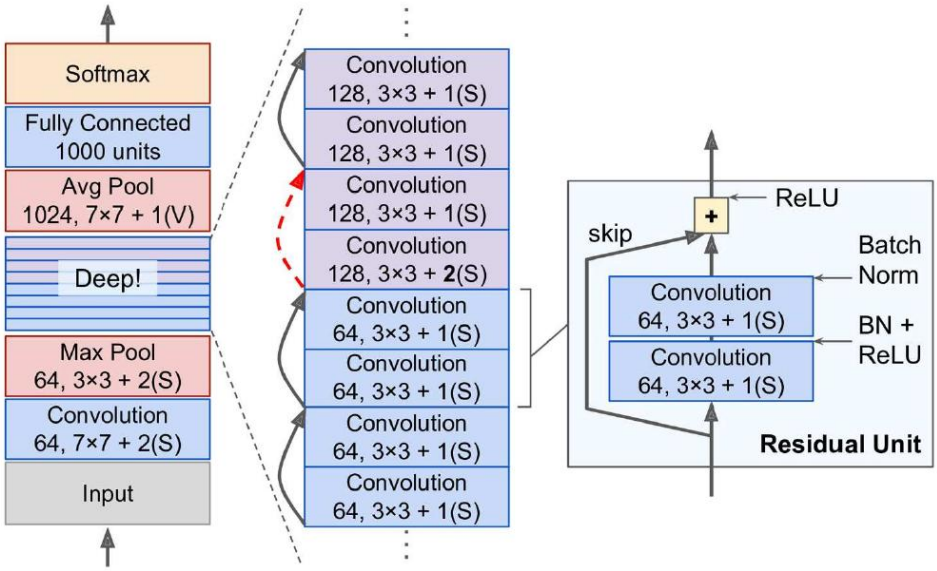

Рисунок 1.3 – Архітектура ResNet

1.1.2 Алгоритми з використанням сингулярного розкладу матриці (SVD)

Перш за все наведемо визначення SVD [3].

Теорема. Якщо матриця $A \in \mathbb{R}^{m \times n}$, то існують ортогональні матриці $U = [u_1, ..., u_m] \in \mathbb{R}^{m \times m}$ та $V = [v_1, ..., v_n] \in \mathbb{R}^{n \times n}$ такі, що

$$A = USV,$$

де $S = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_p)$, $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p \geq 0$ та $p = \min(m, n)$.

Існує багато прикладів використання саме сингулярних значень (далі SV) як вектора ознак для розпізнавання облич. Експерименти, які будуть наведені далі, покажуть, що SV зберігає лише часткову інформацію про зображення, а більша частина її знаходиться в двох ортогональних матрицях U та V.

На рис. 1.4 та 1.5 знаходяться зображення чотирьох різних людей, які підписані A_1, A_2, A_3, A_4. Відповідно до теореми, кожне зображення обличчя може бути подане як $A_i = U_i S_i V_i$, $i = 1, 2, 3, 4$, де S_i – діагональна матриця, яка складається з сингулярних значень. Для того, щоб показати, як заміни матриці SV впливають на зміни зображень замінюємо матриці SV між A_1, A_2, A_3, A_4 відповідно та залишаємо їхні ортогональні матриці незмінними. Оновленні зображення наведено на рис. 1.4 б), г) та рис. 1.5 б), г) відповідно. Ми бачимо, що заміна матриць SV має малий вплив на зовнішність людей, отже, не вона несе в собі багато інформації про зображення. Основна інформація зберігається в ортогональних матрицях U та V. На рис. 1.4 а), в) та 1.5 а), в) знаходяться зображення з такими самими матрицями SV, як на зображеннях рис. 1.4 г), б) та 1.5 г), б) відповідно, але вони характеризують різні зображення.

З прикладів, наведених на рисунках, ми бачимо, що треба використовувати ортогональні матриці U та V, бо вони зберігають основну інформацію про зображення.

а) A_1 б) $U_1S_2V_1$ в) A_2 г) $U_2S_1V_2$

Рисунок 1.4 – Приклад заміни сингулярних значень двох зображень [18]

а) A_1 б) $U_1S_2V_1$ в) A_2 г) $U_2S_1V_2$

Рисунок 1.5 – Приклад заміни сингулярних значень двох зображень [18]

Алгоритм, який використовує описану логіку, реалізований добре відомою бібліотекою OpenCV [16], а саме face_EigenFaceRecognizer(). В ньому використовується SVD та алгоритм PCA для вибору необхідної кількості векторів, які будуть описувати зображення.

PCA (аналіз головних компонент) – це метод лінійного перетворення, що намагається знаходити напрямки з максимальною дисперсією у багатовимірних даних та проектує їх на новий підпростір, розмірність якого менша за початкову. На рис. 1.6 наведена геометрична інтерпретація методу PCA [5] (X_1, X_2 – початкові вісі ознак, PC_1, PC_2 – головні компоненти).

Суть цього алгоритму полягає в тому, що кожне зображення людини, яке подається у вигляді матриці $(n \times n)$, переводиться у вектор довжиною n^2. Потім усі вектори, які відносяться до однієї людини, об’єднуються в матрицю розміром $(k \times n^2)$, де k – кількість зображень однієї людини. За допомогою PCA ро-

змір цієї матриці зменшується. Разом з цим, частина інформації, що несуть вектори, втрачається. Це і є недоліком цього алгоритму.

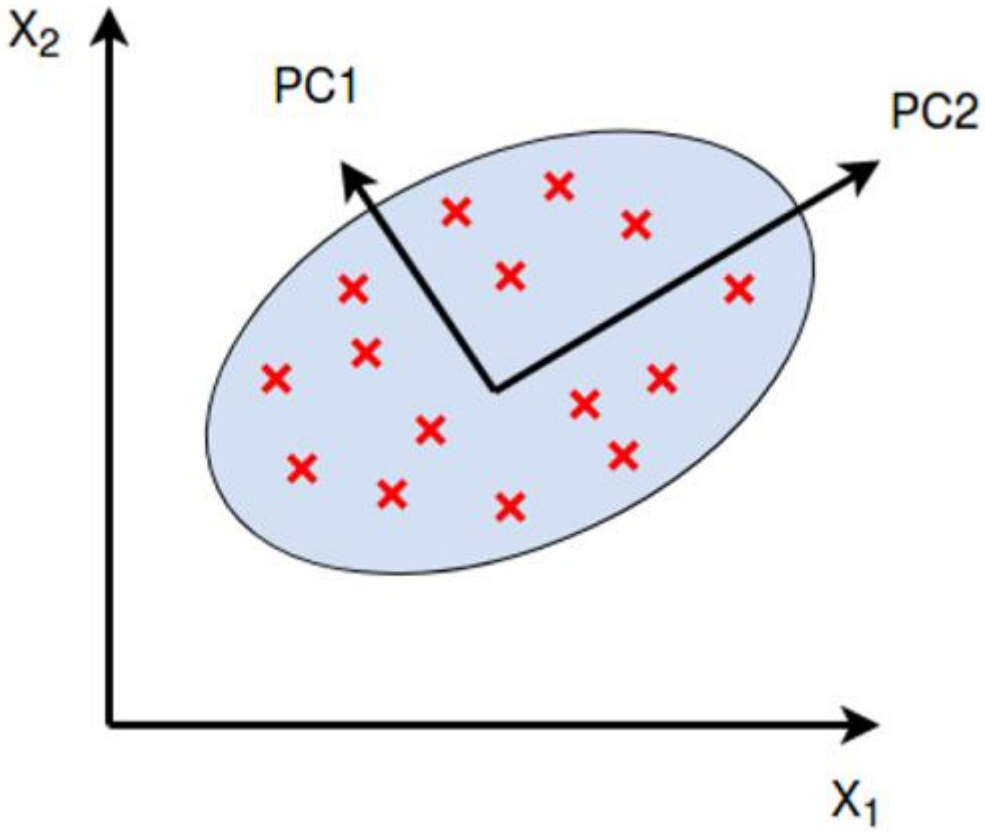

Рисунок 1.6 – Геометрична інтерпретація методу PCA [14]

1.2 Змістовна та формальна постановка задачі

Розглянемо задачу розпізнавання облич. Вона може бути сформульована в термінах задачі бінарної класифікації, де вхідними даними є фотографія (зображення) обличчя людини, а виходом – мітка належності зображення до одного з класів, тобто ця мітка вказує на людину, чиє обличчя зображене на фотографії.

Вхідні дані представлені у вигляді фотографій n людей. Для кожної людини є одинадцять фотографій на яких вона: засвічена світлом прямо, в окулярах, усміхнена, засвічена світлом зліва, без окулярів, спокійна, засвічена світ-

лом справа, засмучена, сонлива, здивована та з сильно зажмуреними очима. Кожна фотографія має розміри (243×320) пікселів.

Перед тим, як переходити до задачі розпізнавання облич, спочатку, необхідно виділити обличчя на фотографіях. Для цього використовується алгоритм CascadeClassifier. Далі зображення облич «вирізаються» з оригінальних зображень та приводяться до одного розміру. Після цього отримуємо зображення розміру (152×152) пікселів. Приклади зображень наведені на рис. 1.7.

Рисунок 1.7 – Приклади зображень облич

Наступним кроком є отримання «середнього» обличчя для кожної людини. Для цього використовується формула:

$$mean\_image = \frac{image}{image\_count\_per\_person}.$$

Приклади середніх облич наведено на рис. 1.8.

Рисунок 1.8 – Приклади середніх облич

Далі за допомогою SVD отримуються вектори, які характеризують кожне обличчя та з якими будуть порівнюватися фотографії у задачі класифікації.

1.3 Постановка задач дослідження

Виходячи з проведеного аналізу алгоритмів розпізнавання облич, можна зробити висновок про перспективність використання алгоритму розпізнавання облич на основі використання SVD розкладу матриці.

Отже, метою кваліфікаційної роботи є розв’язання задачі розпізнавання облич за допомогою алгоритму SVD. Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести огляд і аналіз сучасного стану задачі розпізнавання облич;

– провести огляд основних понять лінійної алгебри, які застосовуються при використанні алгоритму розпізнавання облич на основі SVD;

– застосувати алгоритм розпізнавання облич з використанням SVD;

– скласти алгоритм розв’язання поставленої задачі;

– програмно реалізувати розроблений алгоритм за допомогою мови програмування Python та середовища Jupyter notebook;

– провести низку обчислювальних експериментів для тестових даних.

2 ВИБІР ТА ОБҐРУНТУВАННЯ МЕТОДУ РОЗВ'ЯЗАННЯ

2.1 Векторні простори

Для побудови загальної теорії систем лінійних рівнянь не достатньо лише детермінантів та матриць, які ми використовуємо при розв'язанні систем лінійних рівнянь. Ми маємо використовувати поняття багатовимірного векторного простору [3].

Означення. Сукупність усіх можливих упорядкованих систем з n дійсних чисел після введення на неї операцій додавання та множення на число називається n-вимірним векторним простором V.

Розглянемо інше означення.

Означення. Простір V називається векторним відносно простору дійсних чисел $\mathbb{R}$, якщо він має наступні властивості:

– якщо α та β належать простору V, то їх сума також належить цьому простору:

$$\text{якщо } \alpha,\beta \in V, \text{ то } \alpha+\beta \in V;$$

– додавання векторів α, β та γ, які належать простору V, асоціативне:

$$\alpha+(\beta+\gamma)=(\alpha+\beta)+\gamma;$$

– існує єдиний нульовий вектор 0:

$$\alpha+0=\alpha;$$

– для кожного вектора α існує єдиний протилежний вектор $-\alpha$:

$$\alpha+(-\alpha)=0;$$

– додавання векторів α та β, які належать простору V, комутативне:

$$\alpha + \beta = \beta + \alpha;$$

– множення на число дистрибутивне відносно додавання векторів:

$$c(\alpha + \beta) = c\alpha + c\beta;$$

– множення на вектор дистрибутивне відносно додавання чисел:

$$(c_1 + c_2)\alpha = c_1\alpha + c_2\alpha;$$

– множення на число асоціативне:

$$c_1(c_2\alpha) = (c_1c_2)\alpha;$$

– існує одиничний елемент такий, що:

$$1 \cdot \alpha = \alpha.$$

У лінійному просторі найбільший інтерес мають системи векторів, у вигляді лінійної комбінації яких можна подати будь-який вектор у єдиний спосіб. Якщо зафіксувати таку систему векторів, то будь-який вектор можна буде однозначно виразити через набір чисел, які є коефіцієнтами відповідної лінійної комбінації.

Означення. Базисом лінійного [4] (векторного) простору V називають будь-яку упорядковану систему векторів для якої виконані дві умови:

– ця система векторів лінійно незалежна;

– кожен вектор у лінійному просторі можна подати у вигляді лінійної

комбінації векторів цієї системи.

Нехай $\{\alpha_1,...,\alpha_n\}$ – базис у лінійному просторі V. Тоді будь-який вектор $x \in V$ може бути записаний в такому вигляді:

$$x = x_1\alpha_1 + ... + x_n\alpha_n .$$

Таку форму запису називають розкладанням вектора x за базисом $\{\alpha_1,...,\alpha_n\}$.

Розглянемо деякі властивості базисних векторів [6].

Властивість 1. У лінійному просторі розкладання будь-якого вектору за даним базисом унікальне.

Властивість 2. Якщо $\{\alpha_1,...,\alpha_n\}$ – базис у просторі V, то:

– будь-які $n+1$ елементи з цього простору є лінійно залежними;

– менше ніж n елементів не можуть бути базисом у цьому просторі;

– будь-який набір з n лінійно незалежних векторів або таких, що утворюють лінійну оболонку простору V, є базисом цього простору

2.2 Лінійні перетворення

Нехай є n-вимірний лінійний простір, яке ми позначимо V_n. Розглянемо перетворення цього простору, тобто відображення, яке переводить кожний вектор α простору V_n в деякий вектор α' цього ж простору. Вектор α' називається образом вектора [1] α при розглядуваному перетворенні.

Означення. Перетворення φ лінійного простору V_n називається лінійним перетворенням цього простору, якщо суму двох будь-яких векторів α та β воно переводить в суму образів цих векторів:

$$\varphi(\alpha + \beta) = \varphi(\alpha) + \varphi(\beta),$$

а множення будь-якого вектора α на будь-яке число c переводить у множення образу вектора α на це саме число c:

$$\varphi(c\alpha) = c\varphi(\alpha).$$

З цього означення випливає, що лінійне перетворення лінійного простору переводить будь-яку лінійну комбінацію даних векторів $\alpha_1, \alpha_2, ..., \alpha_k$ у лінійну комбінацію (з тими ж самими коефіцієнтами) образів цих векторів:

$$\varphi(c_1\alpha_1 + c_2\alpha_2 + ... + c_k\alpha_k) = c_1\varphi(\alpha_1) + c_2\varphi(\alpha_2) + ... + c_k\varphi(\alpha_k).$$

Поняття власного вектора та власного значення [6] є одними з ключових у лінійній алгебрі, на їх основі будується безліч конструкцій. Це пов'язано з тим, що багато співвідношень, пов'язаних з лінійними операторами [2], значно спрощуються у системі координат, побудованої на базисі з власних векторів оператора. Множина власних значень лінійного оператора (спектр оператора) характеризує важливі властивості оператора без прив'язки до будь-якої конкретної системи координат.

Власні вектори та власні значення матриці знаходяться з її характеристичного рівняння. Отже, необхідно дати означення характеристичного рівняння.

Для довільної квадратної матриці $A = (a_{ij})$ порядку n розглянемо детермінант:

$$\det(A - \lambda E) = \begin{vmatrix} a_{11} - \lambda & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} - \lambda & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} - \lambda \end{vmatrix},$$

де E – одинична матриця;

λ – числова змінна.

Відносно змінної λ цей детермінант є багаточленом ступеня n та може бути записаним у вигляді:

$$\chi_A(\lambda) = \det(A - \lambda E) = \sum_{k=0}^{n} (-1)^k d_k \lambda^k .$$

Означення. Багаточлен $\chi_A(\lambda) = \det(A - \lambda E)$ називають характеристичним багаточленом матриці A [2], рівняння $\det(A - \lambda E) = 0$ – характеристичним рівнянням матриці A.

Нехай лінійний оператор φ, який задано матрицею A, діє у просторі V. Це означає, що кожному вектору $x \in V$ ставиться у відповідність вектор $y = Ax$ з того самого простору V.

Число λ називається власним значенням [3], а ненульовий вектор x – власним вектором лінійного оператора φ, який задано матрицею A, якщо вони пов'язані між собою співвідношенням:

$$Ax = \lambda x .$$

Якщо x – власний вектор, який відповідає власному значенню λ, то будь-який колінеарний вектор αx при $\alpha \neq 0$ буде теж власним вектором.

Якщо власному значенню λ відповідають два власних вектора x та y, тоді власним вектором також буде будь-який вектор вигляду $\alpha x + \beta y$, де $\alpha\beta \neq 0$.

Теорема. Для того, щоб число λ було власним значенням лінійного оператора, необхідно та достатньо, щоб воно було коренем характеристичного рівняння матриці цього оператора.

Отже, для того щоб знайти власні значення та власні вектори лінійного оператора A, необхідно виконати наступні операції:

– побудувати характеристичне рівняння $\det(A - \lambda E) = 0$ та знайти усі його корені λ_k, які будуть власними значеннями лінійного оператору;

– для кожного власного значення λ_k знайти фундаментальну систему розв'язків однорідної системи лінійних алгебраїчних рівнянь $(A-\lambda_k E)x=0$.

2.3 Евклідів простір

Використання направлених відрізків для зображення сил та переміщень призводить до дуже важливого терміну скалярного добутку векторів.

З фізики відомо, що якщо вектор a означає силу, точка прикладання якої переміщується з початку вектора b в його кінець, то робота ω такої сили визначається як рівність:

$$\omega = |a||b|\cos\{a,b\}.$$

Права частина цієї рівності називається скалярним добутком векторів a, b [5]. Позначають його символом (a,b). Отже,

$$(a,b) = |a||b|\cos\{a,b\}.$$

Таке означення скалярного добутку відноситься лише до ненульових векторів a, b, оскільки тільки для таких векторів визначений кут. Однак, взявши до уваги прообраз скалярного добутку, легко зрозуміти, як його треба довизначити в тому випадку, коли хоча б один з векторів дорівнює нулю. Якщо сила або переміщення задається нульовим вектором, то виконувана робота дорівнює нулю. Тому ми вважатимемо, що $(a,b)=0$, якщо хоча б один з векторів a, b дорівнює нулю.

З формули скалярного добутку випливають наступні геометричні властивості скалярного добутку. Наприклад, кут між двома ненульовими векторами буде гострим (тупим) тоді й тільки тоді, коли скалярний добуток цих векторів

додатній (від'ємний).

Якщо кут між векторами прямий або хоча б один з векторів є нульовим, то скалярний добуток дорівнює нулю. Такі вектори називаються ортогональними.

Ортогональні вектори одиничної довжини називають ортонормованними векторами.

Розглянемо властивості скалярного добутку векторів:

а) $(a,b)=(b,a)$;

б) $(\alpha a,b)=\alpha(a,b)$;

в) $(a+b,c)=(a,c)+(b,c)$;

г) $(a,a)>0$ при $a\neq 0$; $(0,0)=0$.

Теорема. Якщо два вектори a, b задані своїми декартовими прямокутними координатами, то скалярний добуток цих векторів дорівнює сумі попарних добутків відповідних координат.

Почнемо огляд ортогональної системи векторів з введення поняття евклідового простору.

Означення. Лінійний простір E називається евклідовим, якщо кожній парі векторів a, b з E поставлено у відповідність дійсне число (a,b), яке називається скалярним добутком, більш того, виконуються наступні аксіоми:

а) $(a,b)=(b,a)$;

б) $(\alpha a,b)=\alpha(a,b)$;

в) $(a+b,c)=(a,c)+(b,c)$;

г) $(a,a)>0$ при $a\neq 0$; $(0,0)=0$;

для будь-яких векторів a, b, c з E та будь-якого дійсного числа α.

Система векторів називається ортогональною системою, якщо усі вектори цієї системи попарно ортогональні між собою.

Будь-яка ортогональна система ненульових векторів лінійно незалежна.

Опишемо процес ортогоналізації, тобто деякий спосіб переходу від будь-якої лінійно незалежної системи з k векторів

$$a_1,\ a_2,\ ...,\ a_k$$

евклідового простору E_n до ортогональної системи, яка також складається з k ненульових векторів; ці вектори будуть позначені через $b_1,\ b_2,\ ...,\ b_k$.

Покладемо $b_1 = a_1$, тобто перший вектор початкової системи увійде ортогональну систему, яку ми будуємо. Покладемо далі,

$$b_2 = \alpha_1 b_1 + a_2 .$$

Оскільки $b_1 = a_1$, а вектори a_1 та a_2 лінійно незалежні, то вектор b_2 не дорівнює нулю при будь-якому числі α_1. Підберемо це число з умови, що вектор b_2 має бути ортогональним до вектора b_1:

$$0 = (b_1, b_2) = (b_1, \alpha_1 b_1 + a_2) = \alpha_1 (b_1, b_1) + (b_1, a_2),$$

звідки $\alpha_1 = -\dfrac{(b_1, a_2)}{(b_1, b_1)}$. Узагальнюючи отриману формулу, матимемо:

$$\alpha_i = -\frac{(b_i, a_{i+1})}{(b_i, b_i)}.$$

Продовжуючи цей процес, ми побудуємо шукану ортогональну систему $b_1,\ b_2,\ ...,\ b_k$.

Застосовуючи процес ортогоналізації до довільного базису простору E_n, ми отримаємо ортогональну систему з n ненульових лінійно незалежних векторів, яка буде ортогональним базисом.

Твердження. Будь-який евклідовий простір має ортогональні базиси, причому будь-який ненульовий вектор цього простору входить до складу деякого ортогонального базису.

Назвемо вектор b нормованим, якщо його скалярний квадрат дорівнює одиниці, $(b,b)=1$.

Якщо $a\neq 0$, то $(a,a)>0$ і нормуванням вектора a називається перехід до вектора

$$b=\frac{1}{\sqrt{(a,a)}}a.$$

Вектор b буде нормованим, оскільки

$$(b,\ b)=\left(\frac{1}{\sqrt{(a,\ a)}}a,\ \frac{1}{\sqrt{(a,\ a)}}a\right)=\left(\frac{1}{\sqrt{(a,\ a)}}\right)^2(a,\ a)=1.$$

Базис $e_1,\ e_2,...,\ e_n$ евклідового простору E_n називається ортонормованим, якщо він ортогональний, а усі його вектори нормовані, тобто

$$(e_i,e_j)=0 \text{ при } i\neq j,$$
$$(e_i,e_i)=1,\ i=1,\ 2,\ ...,\ n.$$

Твердження. Будь-який евклідовий простір має ортонормований базис.

Базис $e_1,\ e_2,\ ...,\ e_n$ евклідового простору E_n тоді і тільки тоді буде ортонормованим, якщо скалярний добуток будь-яких двох векторів простору дорівнює сумі добутків відповідних координат цих векторів у зазначеному базисі, тобто з

$$a=\sum_{i=1}^{n}\alpha_i e_i,\ b=\sum_{j=1}^{n}\beta_j e_j$$

випливає, що

$$(a,b)=\sum_{i=1}^{n}\alpha_i\beta_i\ .$$

Нехай задане дійсне лінійне перетворення n змінних:

$$x_i=\sum_{k=1}^{n}q_{ik}y_k,\ i=1,2,...,n,$$

матрицю цього перетворення позначимо через Q. Це перетворення переводить суму квадратів змінних $x_1,\ x_2,\ ...,\ x_n$, тобто квадратичну форму $x_1^2+x_2^2+...+x_n^2$, яка є нормальним виглядом додатньо визначених квадратичних форм, в деяку квадратичну форму від змінних $y_1,\ y_2,\ ...,\ y_n$.

Може статися, що ця нова квадратична форма буде сумою квадратів змінних $y_1,\ y_2,\ ...,\ y_n$, тобто може мати місце рівність

$$x_1^2+x_2^2+...+x_n^2=y_1^2+y_2^2+...+y_n^2. \qquad (2.1)$$

Лінійне перетворення, яке має такі властивості, називається ортогональним перетворенням змінних, а його матриця Q – ортогональною матрицею.

Знаючи закон, за яким перетворюється матриця квадратичної форми при виконанні лінійного перетворення змінних, застосовуючи його до нашого випадку та знаючи, що матрицею квадратичної форми, яка є сумою квадратів, є одинична матриця E, ми отримаємо, що рівність (2.1) рівносильна матричній рівності

$$Q^TEQ=Q^TQ=E\ .$$

Звідси

$$Q^T = Q^{-1},$$

а тому має місце рівність:

$$QQ^T = E. \qquad (2.2)$$

Таким чином, ортогональну матрицю Q можна визначити як таку матрицю, для якої транспонована матриця Q^T дорівнює оберненій матриці Q^{-1}.

Оскільки стовпці матриці Q^T є строками матриці Q, то з (2.2) випливає наступне твердження: квадратна матриця Q тоді й тільки тоді буде ортогональною, якщо сума квадратів всіх елементів будь-якої її строки дорівнює одиниці, а сума добутків відповідних елементів будь-яких двох її різних строк дорівнює нулю.

Побудувати ортонормований базис можна, відштовхуючись від деякого початкового базису, за допомогою алгоритму, який називають процесом ортогоналізації Грама-Шмідта.

Нехай $f = (f_1 ... f_n)$ – деякий базис у n-вимірному евклідовому просторі E. Модифікуючи цей базис, ми будемо будувати новий базис $e = (e_1 ... e_n)$, який буде ортонормованим. Послідовно знаходимо вектори g_1 та e_1, g_2 та e_2 і далі за формулами:

$$g_1 = f_1, \qquad e_1 = \frac{g_1}{\|g_1\|};$$

$$g_2 = f_2 - (f_2, e_1)e_1, \qquad e_2 = \frac{g_2}{\|g_2\|};$$

$$g_3 = f_3 - (f_3, e_1)e_1 - (f_3, e_2)e_2, \qquad e_3 = \frac{g_3}{\|g_3\|};$$

$$\ldots, \qquad \ldots;$$

$$g_n = f_n - (f_n, e_1)e_1 - \ldots - (f_n, e_{n-1})e_{n-1}, \qquad e_n = \frac{g_n}{\|g_n\|}.$$

Для обґрунтування алгоритму необхідно показати, що жоден з послідовно знайдених векторів g_i не є нульовим вектором (бо процес обірвався би несвоєчасно) та що усі вектори g_i, $i = \overline{1,n}$, попарно ортогональні. Тоді вектори e_i, $i = \overline{1,n}$, утворюють ортогональну систему, але при цьому норма кожного з цих векторів дорівнює одиниці. Ортогональна система з n ненульових векторів лінійно незалежна, отже, в n-вимірному евклідовому просторі є базисом.

Доведення спирається на метод математичної індукції. Відповідно до цього методу ми доведемо, що для будь-якого k, $k = \overline{1,n}$, вектори e_1, ..., e_k утворюють ортогональну систему та їх довжини дорівнюють одиниці.

Нехай вектори e_1, ..., e_k утворюють ортогональну систему. Знайдемо новий вектор g_{k+1} за формулою:

$$g_{k+1} = f_{k+1} - (f_{k+1}, e_1)e_1 - \ldots - (f_{k+1}, e_k)e_k.$$

Припустимо, що $g_{k+1} = 0$. Тоді

$$f_{k+1} = (f_{k+1}, e_1)e_1 + \ldots + (f_{k+1}, e_k)e_k, \qquad (2.3)$$

тобто вектор f_{k+1} є лінійною комбінацією векторів e_1, ..., e_k, які виражаються через вектори f_1, ..., f_k. З цього випливає, що цей вектор є лінійною комбінацією системи векторів f_1, ..., f_k, а система векторів f_1, ..., f_k, f_{k+1} – лінійно залежна. Але це суперечить умові лінійної незалежності системи f_1, ..., f_n.

Отже, припущення про те, що $g_{k+1} = 0$, призвело до протиріччя, отже воно є невірним. Нам залишається переконатися, що вектор g_{k+1} є ортогональним до кожного з векторів e_1, ..., e_k. Помножимо рівняння (2.3) скалярно на вектор e_i,

де $i \le k$. Враховуючи, що вектори e_j попарно ортогональні при $j \le k$, отримаємо:

$$(g_{k+1}, e_i) = (f_{k+1}, e_i) - (f_{k+1}, e_i)(e_i, e_i) = (f_{k+1}, e_i) - (f_{k+1}, e_i) = 0,$$

оскільки $(e_i, e_i) = 1$. З цього випливає, що вектори e_1, ..., e_k, e_{k+1}, де $e_{k+1} = \frac{g_{k+1}}{\|g_{k+1}\|}$, утворюють ортогональну систему векторів та мають одиничну довжину.

2.4 Сингулярний розклад матриці

Для будь-якої прямокутної матриці A розміру $m \times n$ завжди існує розклад $A = USV$, де U, V – ортогональні матриці, а S – прямокутна діагональна матриця розміру $m \times n$ з невід'ємними елементами на діагоналі [14].

Якщо заданий сингулярний розклад матриці $A = USV$, тоді:

– діагональні елементи матриці S є сингулярними числами матриці A;

– стовпці матриці U утворюють ортонормований базис з власних векторів матриці AA^T;

– стовпці матриці V^T утворюють ортонормований базис з власних векторів матриці $A^T A$;

– стовпці матриць V^T, U утворюють у сукупності сингулярні базиси матриці A.

Якщо заданий сингулярний розклад матриці $A = USV$, тоді через ортогональність матриць U, V отримуємо такі рівності:

$$AA^T(U) = U(\Sigma\Sigma^T), \quad A^T A(V^T) = V^T(\Sigma^T\Sigma).$$

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Основні можливості мови програмування Python та середовища Jupyter Notebook

Розв’язання задачі розпізнавання облич було реалізовано за допомогою мови програмування Python [6]. Python – це скриптова мова, що активно розвивається, яку використовують для розв’язання великого обсягу найрізноманітніших проблем і завдань. Python стане в нагоді у створенні комп’ютерних та мобільних застосунків, його застосовують у роботі з великим обсягом інформації, при розробці web-сайтів та інших різноманітних проектів, використовують у машинному навчанні. Розробка мови Python почалася наприкінці 80-х рр. XX-го століття. Для розподіленої операційної системи «Amoeba» знадобилася скриптова мова, що розширюється, і співробітник голландського інституту Гвідо ван Россум почав писати таку мову у вільний час. Вже у 1991 році Гвідо опублікував перший код.

Розглянемо особливості та переваги мови програмування Python:

– об’єктно-орієнтоване програмування (ООП): реалізація ООП в Python хоч і специфічна порівняно з іншими об’єктно-орієнтованими мовами, але водночас непогано продумана;

– модулі та пакети: програмне забезпечення (ПЗ) на Python оформляється у вигляді модулів, які можуть бути зібрані в пакети;

– мова характеризується логічним синтаксисом, унаслідок чого вихідний код програм, написаних «на Python», легко читається та сприймається.

– ще одна з переваг цієї мови програмування – її умовна легкість: Pythonн вважається найбільш підходящим для фахівців-початківців, бо розробляти нескладні програми можна навчитися вже через пару-трійку днів вивчення;

– велика інтернет-спільнота: якщо розробник стикається з питаннями та труднощами, він завжди може запитати поради у колег, що значно прискорює вирішення проблем;

– гнучкість та масштабованість: Python дозволяє розробникам адаптувати високорівневу логіку програми, що дозволяє легко розширювати складні програми за необхідності;

– Python є інтерпретованою мовою програмування: це означає, що до запуску він є звичайним текстовим файлом, відповідно програмувати можна майже на всіх платформах.

Розглянемо основні недоліки Python:

– швидкість роботи: високопродуктивні проекти на чистому Python написати буде важко, для цього потрібно вдаватися до інших мов;

– безпека, яка забезпечується моделлю пам’яті мови Python, зводить нанівець більшість можливих процесорних оптимізацій.

Jupyter-ноутбук – це середовище розробки, де одразу можна бачити результат виконання коду та його окремих фрагментів. Його відміна від традиційного середовища розробки в тому, що код можна розбити на шматки та виконувати їх у довільному порядку. Уявіть, що ви можете написати шматочок коду на серветці та сказати серветці: «Виконайся».

У такому середовищі розробки можна, наприклад, написати функцію і перевірити її роботу, без запуску програми повністю. А ще можна змінити порядок виконання коду. Можна окремо завантажити файл у пам’ять, окремо перевірити його, окремо обробити вміст.

А ще в jupyter-ноутбуках є висновок результату одразу після фрагмента коду. Наприклад, можна прямо в середині коду побачити побудований графік, отримати попередні цифри чи іншу візуалізацію.

3.2 Алгоритм розв’язання задачі розпізнавання облич

Нехай маємо N людей в базі даних та кожна людина має K тренувальних зображень обличчя ($F_j,\ j = 1, ..., K$). Розглянемо основні кроки алгоритму розпізнавання облич за допомогою SVD.

Крок 1. Усі зображення облич людей мають пройти через попередню обробку, бути нормалізованими та приведеними до одного розміру.

Крок 2. Для людини i усереднюємо тренувальні зображення для того, щоб створити узагальнене зображення обличчя $M_i = \frac{1}{K}\sum_{j=1}^{K} F_j$.

Крок 3. Розкладаємо кожну матрицю M_i за допомогою SVD, отримаємо матриці U_{M_i}, V_{M_i} та Σ_{Mi} і зберігаємо їх як параметри моделі для людини i.

Крок 4. Проектуємо кожне тренувальне зображення *image* обличчя кожної людини i на ортогональні матриці U_{M_i}, V_{M_i} для отримання векторів з коефіцієнтами проекції за формулою:

$$projection_i = diag(U_{M_i}^T \bullet image \bullet V_{M_i}^T).$$

Крок 5. Об'єднуємо кожний спроектований вектор $projection_i$ у матрицю P_i розміром $(n \times w)$, де n – кількість тренувальних зображень обличчя людини i, а w – довжина вектору $projection_i$.

Крок 6. Обчислюємо математичні сподівання та дисперсії μ_i та σ_i^2 для кожної матриці P_i. Отримані вектори, математичні сподівання та дисперсії використовуватимуться для прогнозування належності об'єкта до одного з класів.

Крок 7. Для того, щоб впізнати невідоме зображення обличчя X, проектуємо X на ортогональні матриці кожного відомого обличчя для отримання векторів з коефіцієнтами проекції. Обчислюємо правдоподібність відношення X до одного з відомих класів за формулою

$$L(X \mid \mu_i, \sigma_i^2) = -\log(\sigma_i^2) - \frac{1}{2}\left(\frac{X - \mu_i}{\sigma_i^2}\right)^2.$$

Зображення X буде відноситися до класу з найбільшою правдоподібністю.

3.3 Опис програми

Детально опишемо програмну реалізацію розв'язання проблеми розпізнавання облич.

На рисунку 3.1 перераховані бібліотеки Python [5], які були імпортовані для виконання роботи.

```
import os
import cv2
import tqdm
import skimage
import skimage.io
import numpy as np
from sklearn.model_selection import LeaveOneOut
```

Рисунок 3.1 – Імпортовані бібліотеки Python

На рисунку 3.2 описаний функціонал, який зчитує зображення облич, потім «вирізає» самі обличчя за допомогою функції detectMultiScale(), яка реалізована бібліотекою OpenCV. Ця функція виділяє на зображенні обличчя, потім функція crop() обрізає оригінальне зображження, для того, щоб зменшити обсяг даних, які будуть оброблятися алгоритмом. Змінна mean_shape зберігає інформацію про середній розмір зображення обличчя.

```
# Find and save face images.

from_path = "yalefaces"
dest_path = "faces"

face_types = [
    'glasses', 'happy', 'leftlight',
    'noglasses', 'normal', 'rightlight', 'sad',
    'sleepy', 'surprised', 'wink', 'centerlight',
]

os.makedirs(dest_path, exist_ok=True)
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

def crop(image, x, y, w, h):
    return image[y:y + h, x:x + w]

face_image_shapes = []
for i in tqdm.tqdm(range(1, 16), desc="Face images", unit="person"):
    for face_type in face_types:
        image_name = "subject" + f"{i}".zfill(2) + f".{face_type}"
        image = np.array(skimage.io.imread(os.path.join(from_path, image_name)))
        faces = face_cascade.detectMultiScale(image, 1.1, 4)
        face_image = crop(image, *faces[0])
        face_image_shapes.append(face_image.shape)
        skimage.io.imsave(os.path.join(dest_path, image_name + ".png"), face_image)

mean_shape = np.mean(face_image_shapes, axis=0)
mean_shape = mean_shape.astype('int64')
print(f"Mean bbox shape: {mean_shape}")
```

```
Face images: 100%|██████████| 15/15 [00:02<00:00,  7.30person/s]
Mean bbox shape: [152 152]
```

Рисунок 3.2 – Функція для виділення обличчя

На рисунку 3.3 зображена функція для приведення усіх зображень облич до одного розміру. Цей етап є необхідним, оскільки алгоритм розпізнавання облич працює із зображеннями одного розміру.

```
# Resize saved images to mean width and height.

from_path = "faces"
dest_path = "faces"

for image_name in tqdm.tqdm(os.listdir(from_path), desc="Face images", unit='image'):
    image = np.array(skimage.io.imread(os.path.join(from_path, image_name)))
    resized = cv2.resize(image, tuple(mean_shape), interpolation = cv2.INTER_AREA)
    skimage.io.imsave(os.path.join(dest_path, image_name), resized)
```

```
Face images: 100%|██████████| 165/165 [00:01<00:00, 158.38image/s]
```

Рисунок 3.3 – Функція для приведення усіх зображень облич до одного розміру

На рисунку 3.4 зображена процедура отримання «середніх» зображень облич з попередньою обробкою зображень. Кожне зображення приводиться до однієї гамми, потім усі зображення людини сумуються та діляться на кількість зображень однієї людини.

```
# Pre-process and save mean faces.

from_path = "faces"
dest_path = "mean_faces"

def correct_gamma(image, mid=0.5):
    mean = np.mean(image)
    gamma = np.log(mid*255)/(np.log(mean) + 1e-16)
    corrected = np.power(image, gamma).clip(0,255).astype(np.uint8)
    return corrected

os.makedirs(dest_path, exist_ok=True)
img_cnt_per_person = len(face_types)

for i in tqdm.tqdm(range(1, 16), desc="Face images", unit="person"):
    mean_image = 0
    for face_type in face_types:
        image_name = "subject" + f"{i}".zfill(2) + f".{face_type}.png"
        image = cv2.imread(os.path.join(from_path, image_name), cv2.IMREAD_GRAYSCALE)

        image = correct_gamma(image)
        mean_image += (1. / img_cnt_per_person) * image
    skimage.io.imsave(os.path.join(dest_path, "subject" + f"{i}".zfill(2) + ".png"), np.uint8(mean_image))
```

```
Face images: 100%|██████████| 15/15 [00:00<00:00, 23.28person/s]
```

Рисунок 3.4 – Функція отримання «середніх» зображень облич

Наступним кроком необхідно розкласти отримані матриці «середніх» облич за допомогою алгоритму SVD. Перед цим кожна матриця нормалізується

та після розкладу отримані ортогональні матриці U, V зберігаються у змінну face_uv_db. Цей процес зображений на рисунку 3.5.

```python
# Do SVD on mean faces and store matrices U and Vh into a dictionary.

from_path = "mean_faces"
face_uv_db = {}

def normalize(image):
    return (image / 255).astype(np.float64)

for image_name in tqdm.tqdm(os.listdir(from_path), desc="Persons", unit="image"):
    image = cv2.imread(os.path.join(from_path, image_name), cv2.IMREAD_GRAYSCALE)
    image = normalize(image)
    u, s, vh = np.linalg.svd(image, full_matrices=True)
    face_uv_db[image_name.split('.')[0]] = {'u': u, 'vh': vh}
```

```
Persons: 100%|██████████| 15/15 [00:00<00:00, 179.05image/s]
```

Рисунок 3.5 – Розклад «середніх» облич за допомогою SVD

На рисунку 3.6 наведено функцію, яка проектує зображення на ортогональні матриці U, V та отримує вектори, які є основою для майбутніх передбачень. Також тут знаходяться значення математичного сподівання та дисперсії з матриці, утвореної отриманими векторами, для кожної людини.

```python
# Compute dist params of features on training set

from_path = "faces"

train_face_types = [
    'glasses', 'happy', 'leftlight',
    'noglasses', 'normal', 'sad',
    'sleepy', 'surprised', 'wink','centerlight'
]
test_face_type =  'rightlight'

def get_feature_vector(image, u, vh):
    return np.diag(u.T @ image @ vh.T)

persons_dist_params = {}
for i in tqdm.tqdm(range(1, 16), desc="Face images", unit="person"):
    person_key = "subject" + f"{i}".zfill(2)
    person_embs = []
    for face_type in train_face_types:
        image_name = person_key + f".{face_type}.png"
        image = cv2.imread(os.path.join(from_path, image_name), cv2.IMREAD_GRAYSCALE)
        image = correct_gamma(image)
        image = normalize(image)
        uv = face_uv_db[person_key]
        emb = get_feature_vector(image, uv['u'], uv['vh'])
        person_embs.append(emb)
    person_embs = np.array(person_embs)
    person_mean_svs, person_std_svs = np.mean(person_embs, axis=0), np.std(person_embs, axis=0)
    persons_dist_params[person_key] = {'mu': person_mean_svs, 'sigma': person_std_svs}
```

```
Face images: 100%|██████████| 15/15 [00:00<00:00, 64.27person/s]
```

Рисунок 3.6 – Функція для отримання проекцій зображень

На рисунку 3.7 зображена функця для розрахунку правдоподібності та функція, яка робить передбачення класу, до якого належить зображення (чиє

обличчя є на зображенні?).

```python
def log_proba(emb, mu, sigma):
    probas = - np.log(sigma) - 0.5 * ((emb - mu) / (sigma + 1e-16))**2
    return np.sum(probas)


def predict_person_key(image, face_uv_db, persons_dist_params):
    lgp_max, pred_person_key = -np.inf, None
    for person_key in face_uv_db:
        emb = get_feature_vector(image, **face_uv_db[person_key])
        lgp = log_proba(emb, **persons_dist_params[person_key])
        if lgp > lgp_max:
            lgp_max = lgp
            pred_person_key = person_key
    return pred_person_key
```

Рисунок 3.7 – Функція для отримання проекцій зображень

На рисунку 3.8 зображений алгоритм, який об’єднує в собі функції, які описані вище та робить передбачення о належності зображення до одного з класів. Для обчислювального експерименту дані представлені у вигляді фотографій п’ятнадцяти людей. Для кожної людини є одинадцять різних фотографій людини, на яких вона: засвічена світлом прямо, в окулярах, усміхнена, засвічена світлом зліва, без окулярів, спокійна, засвічена світлом справа, засмучена, сонлива, здивована та з сильно зажмуреними очима. За допомогою алгоритму LeaveOneOut() [15], який реалізований бібліотекою sklearn [15], ми робимо один з типів зображення тестовим (усі зображення людей цього типу будуть тестовими), а усі інші типи – тренувальними. В результаті ми отримуємо точності алгоритму розпізнавання облич при різних тренувальних та тестових наборах.

```python
loo = LeaveOneOut()
for train_face_types_id, test_face_type_id in loo.split(all_face_types):
    train_face_types = all_face_types[train_face_types_id]
    test_face_type = all_face_types[test_face_type_id][0]
    persons_dist_params = get_persons_dist_params(train_face_types, face_uv_db, from_path)
    total = 15
    score = 0.
    for i in range(1, 16):
        person_key = "subject" + f"{i}".zfill(2)
        image_name = person_key + f".{test_face_type}.png"
        image = cv2.imread(os.path.join(from_path, image_name), cv2.IMREAD_GRAYSCALE)
        image = correct_gamma(image)
        image = normalize(image)
        pred_person_key = predict_person_key(image, face_uv_db, persons_dist_params)
        score += int(pred_person_key == person_key)
    score /= total
    print(f"Test face type: '{test_face_type}' -- Accuracy: {score*100:.2f}%")
```

Рисунок 3.8 – Алгоритм розпізнвання облич

4 РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРИМЕНТУ ТА ЇХ АНАЛІЗ

Обчислювальний експеримент було проведено для даних представлених у вигляді фотографій п’ятнадцяти людей. Для кожної людини є одинадцять різних фотографій людини, на яких вона: засвічена світлом прямо, в окулярах, усміхнена, засвічена світлом зліва, без окулярів, спокійна, засвічена світлом справа, засмучена, сонлива, здивована та з сильно зажмуреними очима.

Один з типів зображення є тестовим (усі зображення людей цього типу будуть тестовими), а усі інші типи – тренувальними.

На рис. 4.1 наведено приклад типів зображень для однієї людини, а на рис. 4.2 наведено його усереднене зображення.

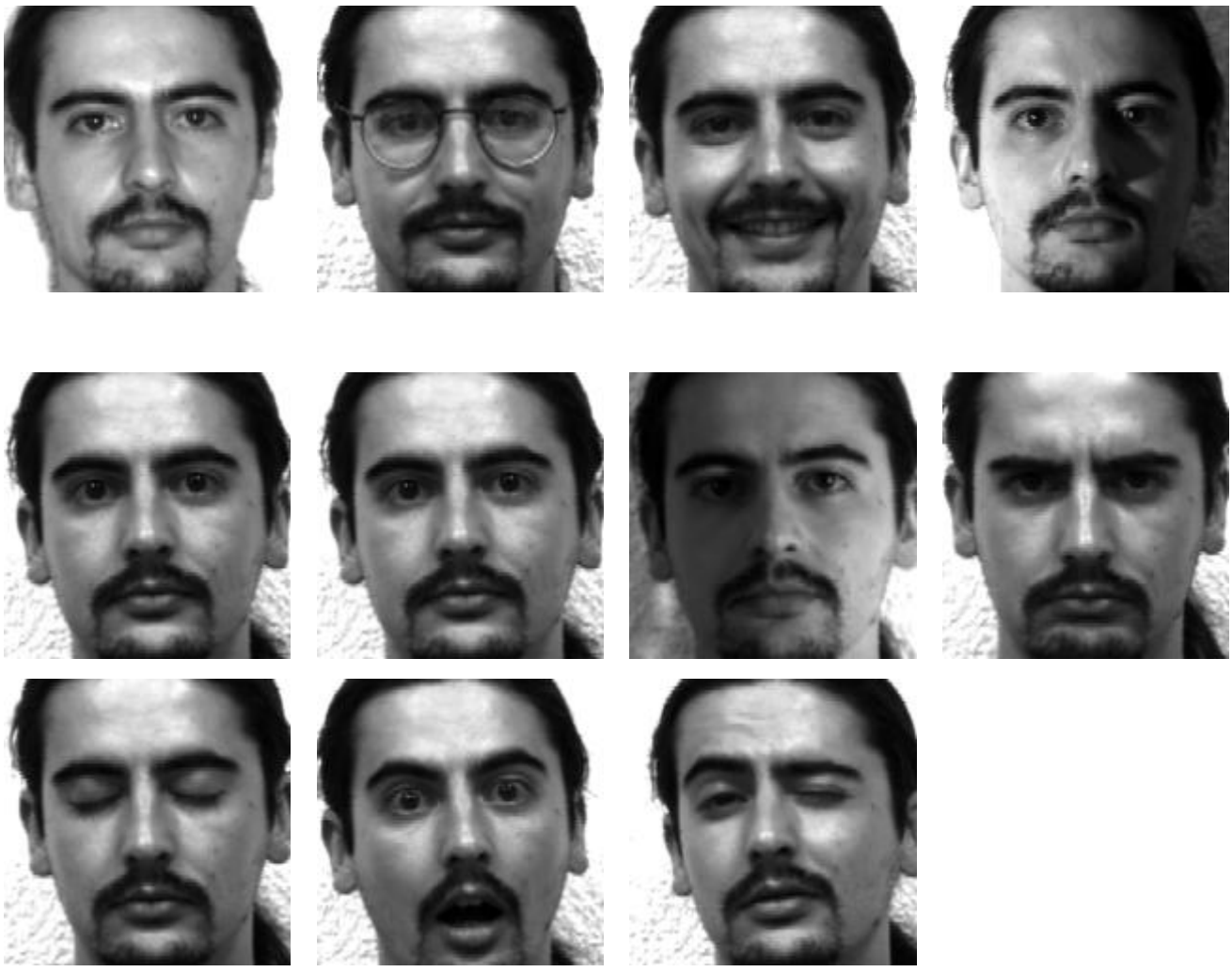

Рисунок 4.1 – Типи зображення облич

Рисунок 4.2 – Усереднене зображення

Дуже важливим етапом в роботі з даними є оцінка роботи алгоритму машинного навчання, оскільки уся попередня робота може не мати сенсу, якщо результати виявляться незадовільними.

В цілому, задачу розпізнавання облич можна звезти до задачі бінарної класифікації з двома можливими класами – об’єкт належить до класу, або не належить. В нашому наборі даних кількість об’єктів однакова для кожного класу, отже можна використовувати класичну метрику для оцінки роботи алгоритму, а саме Accuracy.

$$ACC = \frac{Кількість\,правильних\,передбачень}{Загальна\,кількість\,передбачень} = \frac{TP + TN}{TP + TN + FP + FN},$$

де TP (True Positive) – кількість об’єктів, для яких алгоритм правильно спрогнозував належність до класу 1;

FP (False Positive) – кількість об’єктів, для яких алгоритм неправильно спрогнозував належність до класу 1;

TN (True Negative) – кількість об’єктів, для яких алгоритм правильно спрогнозував належність до класу 0;

FN (False Negative) – кількість об’єктів, для яких алгоритм неправильно спрогнозував належність до класу 0.

Оскільки ми використовуємо алгоритм перемішування даних, завдяки якому кожен з типів зображення буде тестовим, проведемо оцінку роботи моделі з різними наборами тестових даних. В таблиці 4.1 наведені ці результати оцінки.

Таблиця 4.1 – Оцінка роботи моделі

Тип тестових даних	Accuracy
glasses	93,33%
happy	100%
left light	100%
no glasses	100%
normal	100%
right light	86,67%
sad	100%
sleepy	100%
surprised	100%
wink	100%
center light	100%
Average	98,15%

Отримані результати, які представлені в таблиці 4.1, є досить високими з огляду на початкову складність задачі. Звісно, результати роботи алгоритму можна покращити. Для можна експериментувати з попередньою обробкою даних. Саме це є особливістю алгоритмів машинного навчання – можна нескінченно перебирати можливі налаштування алгоритму та експериментувати з попередньою обробкою. Зупинятися треба при досягнення необхідної точності, яку ми досягли у розглянутому прикладі роботі.

Результати кваліфікаційної роботи слугують доказом того, що алгоритм класифікації облич з використанням SVD – це потужний алгоритм розпізнавання. Його можна широко застосовувати, якщо в систему розпізнавання облич

будуть приходити однотипні зображення високої якості. Гарним прикладом можуть бути системи верифікації облич на вході в приміщення, при розблокування мобільного телефону, або ноутбуку та інші.

На відміну від більш складних та важких алгоритмів, розпізнавання облич з використанням SVD не потребує обчислюваної системи великої потужності, оскільки в основі цього алгоритму лежать нескладні, з точки зору розрахунків, векторні операції. Завдяки цим властивостям, цей алгоритм дійсно можна широко застосовувати у багатьох побутових галузях, навіть з використанням «мініі» комп’ютерів, або досить застарілих пристроїв.

ВИСНОВКИ

У роботі був проведений огляд методів розв’язання задачі розпізнавання облич.

1. На основі порівняння альтернативних алгоритмів порівняння облич було встановлено, що алгоритм на основі SVD є актуальним та є усі підстави використовувати саме його для розв’язання поставленої задачі.

2. Була розглянута теорія, яка є необхідною для побудови системи розпізнавання облич з використанням алгоритму SVD.

3. Побудовано алгоритм розпізнавання облич з використанням SVD розкладання матриці. Алгоритм був реалізований з використанням мови програмування Python та середовища Jupyter Notebook. Такі бібліотеки як OpenCV та sklearn дозволили виконати попередню обробку даних для того, щоб використовувати їх для навчання на тренувальних даних, які були отримані в результаті розділення вхідного набору даних на тренувальні та тестові. Далі було отримане передбачення моделі на тестових даних. Отриманий алгоритм є точним та здатним виконувати розпізнавання облич. Головною особливістю цього алгоритму є те, що він набагато швидший за альтернативні алгоритми та не потребує великого обсягу розрахункових потужностей.

4. Розроблений алгоритм можна широко застосовувати, якщо в систему розпізнавання облич будуть приходити однотипні зображення високої якості. Гарним прикладом можуть бути системи верифікації облич на вході в приміщення, при розблокування мобільного телефону або ноутбуку та інші.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Воеводин В. В. Линейная алгебра. 2-е изд. Москва : Наука, 1980. 401 с.

2. Воеводин В. В., Воеводин Вл. В. Энциклопедия линейной алгебры. Санкт-Петербург : БХВ-Петербург, 2006. 545 с.

3. Канатников А. Н., Крищенко А. П. Аналитическая геометрия. 2-е изд. Москва : МГТУ им. Н. Э. Баумана, 2000. 388 с.

4. Канатников А. Н., Крищенко А. П. Аналитическая геометрия. 3-е изд. Москва : МГТУ им. Н. Э. Баумана, 2002. 336 с.

5. Курош А. Г. Курс высшей алгебры. Москва : Наука, 1968. 431 с.

6. Орельен Ж. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. Концепции, инструменты и техники для создания интеллектуальных систем. Москва : Вильямс, 2018. 688 с.

7. Рамальо Л. Python. К вершинам мастерства. Москва : ДМК Пресс, 2016. 768 с.

8. Рашка С. Python и машинное обучение. Москва : ДМК Пресс, 2017. 418 с.

9. Pentland A., Choudhury T. Face recognition for smart environments // IEEE Comput. 2000. 33 (2). P. 50–55.

10. Deep Residual Learning for Image Recognition / K. He, X. Zhang, S. Ren, J. Sun // arXiv preprint arXiv:1512.03385v1. 44 p.

11. Zhang J., Yan Y., Lades M. Face recognition: eigenface, elastic matching, andneural nets // Proceedings of the IEEE. 1997. 85 (9). P. 1423–1435.

12. Lutz M. Learning Python. Sebastopol : O`Reilly, 2013. 1542 p.

13. Turk M.A., Pentland A.P. Face recognition using eigenfaces // Proceedings of the International Conference on Pattern Recognition. 1991. Pp. 586–591.

14. Obidinnu J. N. Standardized templates for verifying the syntax-correctness of data flow diagrams // Journal of Science, Engineering and Technology. 2019. V. 6, № 2. P. 9–14.

15. Paper D., Paper D. Introduction to Scikit-Learn // Hands-on Scikit-Learn for Machine Learning Applications: Data Science Fundamentals with Python. 2020.

P. 1–35.

16. Viola P., Jones M. Rapid Object Detection using a Boosted Cascade of Simple Features // Accepted conference on computer vision and pattern recognition. 2001. P. 53–58.

17. Conceptual Understanding of Convolutional Neural Network / S. Indolia, A. K. Goswami, S. P. Mishra, P. Asopa // A Deep Learning Approach, International Conference on Computational Intelligence and Data Science (ICCIDS 2018). P. 65.

18. Towards CRISP-ML (Q): A Machine Learning Process Model with Quality Assurance Methodology / S. Studer, T.B. Bui, C. Drescher and other // arXiv preprint arXiv:2003.05155. 2020. 18 p.

19. Do singular values contain adequate information for face recognition? / Y. Tian, T. Tan, Y. Wang, Y. Fang // National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, Beijing, POBox 2728, China. 2002. 38 p.