КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

«Прикладне програмування»
(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Веб-застосунок розпізнавання обличчя»

Виконав _____ _____
(Підпис)
Касьяненко Данило Михайлович
(прізвище, ім'я, по батькові)

Керівник _Плескач Валентина Леонідівна
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист:

_____

_____
(Висновок: "До захисту в екзаменаційній комісії ")

Завідувач кафедри ____ _____ Плескач В.Л. _____
(Підпис) (Прізвище, ініціали) (Дата)

Засвідчую, що у цій дипломній роботі немає запозичень із праць інших авторів без відповідних посилань.

Унікальність тексту - 95 %

Київ – 2022

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра прикладних інформаційних систем

Назва теми: «Веб-застосунок розпізнавання обличчя»

Освітня програма: Прикладне програмування

Спеціальність: Комп’ютерні науки

ПІБ	Підпис
Касьяненко Данило Михайлович	

Назва роботи українською та англійською мовами:

Веб-застосунок розпізнавання обличчя
Face recognition web application

Мета бакалаврської роботи: Швидке розпізнавання обличчя на основі створеного прототипу веб-застосунку.
План роботи: 1. Теоретичні основи аналізу зображень облич на основі нейромереж 2. Аналіз архітектурних рішень і вибір програмних засобів для створення веб-сервісу 3. Проектування, реалізація і впровадження програмної системи розпізнавання облич

ПІБ, ступінь, звання наукового керівника роботи:

Плескач Валентина Леонідівна д.е.н., к.т.н., проф.

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	09.10.2021	виконано
2.	Видача завдання кваліфікаційної роботи бакалавра	19.10.2021	виконано
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	21.10.2021	виконано
4.	Затвердження плану кваліфікаційної роботи бакалавра	25.10.2022	виконано
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	01.11.2022	виконано
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	21.12.2022	виконано
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	31.01.2022	виконано
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	30.03.2022	виконано
9.	Подання роботи у першому варіанті	28.04.2022	виконано
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	03.05.2022	виконано
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**23.05.2022**	Виконано
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	27.05.2022	Виконано
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	10.06.2022	виконано
14.	Захист кваліфікаційної роботи бакалавра	22.06.2022 23.06.2022 24.06.2022	

Здобувач вищої освіти  (підпис)

Керівник  (підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Завдання до дипломної роботи(календарний план дипломної роботи)	1
Відомість дипломної роботи	1
Анотація	1
Анотація (іноземною мовою-англійською)	1
Зміст	1
Перелік скорочень, умовних позначень, термінів	1
Вступ	1
Розділ 1	15
Розділ 2	13
Розділ 3	14
Висновок	1
Перелік використаних джерел	7
Додатки	12

				ДП ХХХХ 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.	Касьяненко Д.М..			Відомість дипломної роботи	Лист	Листів
Керівн.	Плескач В.Л.				1	73
Н/контр.	Базилюк А.М.					
Зав.каф.	Плескач В.Л.					

Пояснювальна записка

до дипломної роботи

на тему: «Веб-застосунок розпізнавання обличчя» полягає в здійсненні дослідження рівня розвитку технологій розпізнавання обличчя; аналізу теоретичних основ обробки зображень облич на основі нейромереж та програмно-технологічних рішень для створення веб-застосунків; проектування, реалізації, тестуванні та впровадження веб-застосунку розпізнавання обличчя.

Київ-2022

АНОТАЦІЯ

Дипломна робота: 73 с., 2 рис., 99 джерел, 5 дод.

Цю кваліфікаційну роботу присвячено проектуванню та розробленню веб-застосунку розпізнавання обличчя.

Метою дипломної роботи є швидке розпізнавання обличчя на основі створеного прототипу веб-застосунку.

Для досягнення поставленої мети потрібно вирішити такі **завдання**:

1. Дослідити теоретичні основи аналізу зображень облич на основі нейромереж.
2. Проаналізувати архітектурні рішення і обрати програмні засоби для створення веб-сервісу.
3. Спроектувати, реалізувати та впровадити програмну систему розпізнавання облич.

Об’єкт дослідження.

Процес розпізнавання обличчя.

Предмет дослідження.

Засоби і принципи побудови програмної системи розпізнавання обличчя.

Методи дослідження.

Метод аналізу та синтезу, метод індукції та дедукції, системний метод, математичний метод, статистичний метод, емпіричний метод.

Ключові слова: розпізнавання облич, веб-застосунок, бази даних, нейромережі.

.

ABSTRACT

Thesis: 73 pages, 2 figures, 99 sources, 5 appendices

This thesis is devoted to the design and development of face recognition web application.

The purpose of the thesis is fast face recognition for web application.

In order to achieve this goal you need to solve the following **tasks**:

1. To study the general theoretical foundations of image analysis using neural networks;
2. To carry out the analysis of software and technological decisions of construction of web applications;
3. To design and implement a face recognition system.

The object of research.

Face recognition processes.

Subject of research.

Software and technical principles, approaches to creating face recognition systems.

Research methods.

Analytic–synthetic method, deductive and inductive reasoning, systems thinking, mathematical method, statistical method, empirical method.

Keywords: face recognition, web application, databases, neural networks.

ЗМІСТ

ВСТУП **9**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ **10**

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ АНАЛІЗУ ЗОБРАЖЕНЬ ОБЛИЧ НА ОСНОВІ НЕЙРОМЕРЕЖ

1.1 Основи теорії штучного інтелекту при аналізі зображень 11

1.2 Класичні методи розпізнавання облич 19

1.3 Новітні методи та системи штучного інтелекту розпізнавання облич 23

РОЗДІЛ 2. АНАЛІЗ АРХІТЕКТУРНИХ РІШЕНЬ І ВИБІР ПРОГРАМНИХ ЗАСОБІВ ДЛЯ СТВОРЕННЯ ВЕБ-СЕРВІСУ

2.1 Фреймворк Django для створення веб сайтів на Python 26

2.2 Вибір програмної реалізації для модулю розпізнавання облич 31

2.3 Переваги і недоліки алгоритмів систем розпізнавання облич 36

РОЗДІЛ 3. ПРОЕКТУВАННЯ, РЕАЛІЗАЦІЯ І ВПРОВАДЖЕННЯ ПРОГРАМНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ОБЛИЧ

3.1 Постановка задачі 39

3.2 Реалізація програмної системи розпізнавання облич 42

3.3 Тестування і впровадження системи розпізнавання облич 48

ВИСНОВОК **52**

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **53**

ДОДАТКИ **61**

ВСТУП

Об’єкт дослідження: процес розпізнавання обличчя.

Предмет дослідження: засоби і принципи побудови програмної системи розпізнавання обличчя.

Мета: швидке розпізнавання обличчя на основі створеного прототипу веб-застосунку.

Завдання:

1) Дослідити теоретичні основи аналізу зображень облич на основі нейромереж.
2) Розкрити основи теорії штучного інтелекту при аналізі зображень.
3) Визначити відомі методи та системи штучного інтелекту розпізнавання облич.
4) Здійснити аналіз програмно-технічних рішень розпізнавання облич.
5) Розглянути фреймворк Django для створення веб сайтів на Python.
6) Оцінити можливість використання проекту моделей для розпізнавання облич InsightFace.
7) Перерахувати переваги і недоліки алгоритмів систем розпізнавання облич.
8) Спроектувати, реалізувати та впровадити програмну систему розпізнавання облич

Використані методи: метод аналізу та синтезу, метод індукції та дедукції, системний метод, математичний метод, статистичний метод, емпіричний метод.

Робота складається з трьох розділів, вступу, списку використаних джерел.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ЛДА – лінійний дискримінантний аналіз

MVT – Model View Template

MVC – Model View Control

РОЗДІЛ 1

ТЕОРЕТИЧНІ ОСНОВИ АНАЛІЗУ ЗОБРАЖЕНЬ ОБЛИЧ НА ОСНОВІ НЕЙРОМЕРЕЖ

1.1 Основи теорії штучного інтелекту при аналізі зображень

З часів появи нейронних мереж у цифровій обробці зображень широко використовуються статистичні методи розпізнавання зображень. Спочатку проблеми розпізнавання зображень часто вирішувалися за допомогою лінійних та квадратичних дискримінантів [1], непараметричних методів для k-найближчих сусідів та оцінки щільності Парзена [2, 3]. У середині вісімдесятих років був введений алгоритм навчання нейронних мереж із зворотним поширенням помилок [4]. Цей алгоритм дає змогу навчати глибокі нейронні мережі, тобто нейронні мережі з проміжними шарами. Вони виконують операції над вхідними даними, які важко інтерпретувати людині, але ефективні для підвищення точності моделі. Через це такі шари ще називають «прихованими». Відтоді нейронні мережі з одним або кількома прихованими шарами теоретично можна навчити виконувати більше завдань регресії або класифікації. Крім того, нейронні мережі не роблять припущень щодо статистичного розподілу вхідних даних.

Основні застосування нейронних мереж релевантні для нашого дослідження:

- Стиснення даних і вилучення ознак

Ознаки — це набір чисел, які представляють характеристики, які можна використовувати для ідентифікації об’єкта. Двома найважливішими застосуваннями скорочення даних є стиснення зображень і вилучення ознак. Як правило, алгоритм стиснення зображень для зберігання та передачі зображень складається з двох кроків: кодування та декодування. Видалення об’єктів використовується для подальшого сегментування або ідентифікації об’єктів. Характеристики, які модель видаляє із зображення, зазвичай відповідають конкретним геометричним або перцептивним характеристикам зображення (краї, кути та з’єднання) або специфічним для області застосування моделі, наприклад рисам обличчя.

- Застосування для вилучення ознак

Вилучення ознак можна розглядати як особливий тип скорочення даних, метою якого є пошук підмножини інформативних змінних на основі даних зображення. Оскільки ці зображення дуже великі за своєю природою, видалення ознак часто є необхідним кроком для успішної сегментації або розпізнавання об’єктів. Окрім зниження обчислювальних витрат, вибір функцій також є засобом контролю так званого «прокляття розмірності». При використанні в якості вхідних даних для наступного алгоритму сегментації необхідно визначити ті ознаки, які найкраще зберігають розділення класів [2, 3].

Існує багато типів нейронних мереж, які можна навчити виконувати відображення в низьковимірні простори [5]. Добре відомим прикладом нейронної мережі для виділення ознак є нейронна реалізація одновимірного аналізу головних компонентів, пізніше розширеного до кількох вимірів [6, 7]. Також, було визначено, що навчання тришарової асоціативної мережі еквівалентно застосуванню методу головних компонентів до вхідних даних [8]. Пізніше було показано, що мережі з п’ятьма шарами здатні виконувати нелінійне зменшення розмірності (тобто знаходити головні поверхні) [9, 10, 11]. Ще одним значимим відкриттям є можливість апроксимувати нелінійні підпростори, використовуючи набір лінійних підпросторів [12].

Більшість штучних нейронних мереж, навчених для вилучення ознак, отримують вхідні дані у форматі повного списку пікселів зображення.

Вилучення ознак за допомогою нейронної мережі використовується для:

- подальша автоматична ідентифікація об'єктів інтересу [13] та ідентифікаційних символів [14, 15];

- подальша сегментація оптичних зображень [16] та магнітно-резонансних зображень [17];
- щоб знайти кут повороту об’єкта [18, 19];
- знайти контрольні точки деформованої моделі [20];
- кластеризація низькорівневих ознак, виявлених фільтрами габора при розпізнаванні облич і виявленні дефектів зображення [21];
- порівняння об’єктів між зображеннями, зробленими з різних ракурсів [22];
- вміст локального зображення групується перед кодуванням [23].

У більшості випадків вилучені ознаки використовуються для сегментації, зіставлення зображень або розпізнавання об’єктів. Для об’єктів, які з’являються в одному масштабі, обертання призводить до найбільшої зміни в класі. Деякі методи вибору функцій були спеціально розроблені для роботи з об’єктно-орієнтованими змінами.

Важливо розрізняти використання нейронної мережі натренованої з вчителем і нейронної мережі натренованої без вчителя для вилучення ознак. Для нейронних мереж, які навчаються з вчителем, втрата даних через стиснення даних може бути безпосередньо виміряна за прогнозованими вихідними змінними, які не можуть бути оцінені методами навчання без вчителя. Методи нейронних мереж мають переваги перед традиційними методами, такими як аналіз головних компонентів. Нейронні мережі з прихованими шарами здатні виконувати нелінійне виділення ознак, але не мають формальної статистичної бази, яка б якимось чином гарантувала їх корисність. Однак на практиці репрезентативность ознак, вилучених нейронними мережами, часто вища, ніж у класичних методів. Вивчення цього явища є одним із головних завдань сучасної науки про дані.

- Розпізнавання об’єктів

Розпізнавання об’єктів включає визначення положення, орієнтації та масштабу об’єктів на зображенні. Може бути ще один крок - визначення класу

виявленого об'єкта. Огляд наукових статей про розпізнавання об’єктів за допомогою штучних нейронних мереж показує, що в більшості програм нейронних мереж їх навчають знаходити окремі об’єкти безпосередньо на основі піксельних даних. Іншим, менш поширеним підходом, є відображення вмісту зображення до простору ознак як вхідних даних для нейронного класифікатора.

- Розпізнавання об'єктів на основі піксельних даних

Серед методів навчання нейронних мереж, спрямованих на розпізнавання об'єктів з пікселів [12, 13, 19, 24, 25, 26, 27], можна виділити такі типи нейронних мереж: прямі нейронні мережі [19, 28, 29, 30]. , 31, 32, 33, 34], нейронні мережі з використанням загальних параметрів [35, 36, 37], рекурентні мережі [27], мережі теорії адаптивного резонансу [24, 38], еволюційні штучні нейронні мережі [40] , двонаправлені цикли [41], неокогнітрон [42, 43] та його варіанти [44, 45], лінійні нейронні класифікатори [46], нейронні мережі вищого порядку [47, 48] та нейронні мережі Хопфілда [49, 50] , 51]. Крім того, існують також апаратні нейронні мережі для розпізнавання об’єктів: «нейронні мережі без параметрів» [52, 53] та оптичні реалізації нейронних мереж [54, 55]. Самоорганізуюча карта Кохонена іноді використовується для вилучення карт ознак із піксельних даних [33, 56]; вихід карти потім подається на вхід нейронної мережі.

Кілька нових мережевих архітектур були спеціально розроблені, щоб впоратися зі змінами положення об’єкта, обертання, масштабу та освітлення. Методи зазвичай спеціалізуються на конкретних типах вхідних зображень. Основними напрямками є розпізнавання двовимірних проекцій і перспектив, проте деякі методи підтримують роботу і з тривимірними даними. Нейронні фільтри [38] є цікавим підходом, який виконує розпізнавання об’єктів і не залежить від 2D зміщення, обертання площини та масштабування. Він поєднує в собі набір фільтрів різного масштабу та орієнтації з блоком відповідності. Інші підходи спираються на вивчення варіації через навчання [57, 28, 58, 31]. Для розпізнавання також можуть використовуватися статистичні моделі об'єктів, які підлягають ідентифікації [28,

58]. Згорткові шари нейронних мереж досліджуються за допомогою синтетичних зображень змодельованих об'єктів із випадково обраними орієнтаціями. Для навчання нейронної мережі можна використовувати також штучні дані, згенеровані за допомогою комп'ютеру. Наприклад, такий підхід був використаний для виявлення вузлів легенів на рентгенограмах грудної клітки [31]. Ці нейронні мережі частково вивчаються за допомогою синтетичних зображень легеневих вузлів і частково за допомогою реальних знімків. Інші розробили методи, які є інваріантними як до зміщення, так і до повороту в площині зображення [35, 59], або системи, які виконують інваріантну обробку подібним чином зміщення та/або в різних масштабах завдяки своїй архітектурі. Наприклад, неокогнітрон [42] та мережі з повторним використанням параметрів [60, 36]. Також деякі використовують ієрархічний підхід для інваріантного обертанням розпізнавання об'єктів [19].

Очевидно, що коли розпізнавання об’єктів виконується шляхом навчання класифікатора ідентифікувати весь об’єкт за необробленими значеннями всіх пікселів зображення, складність класифікатора значно зростає з розміром об’єкта та кількістю вимірювань. Цікавим способом обійти цю проблему є ітераційний пошук центру об’єкта на зображенні [52]. У цьому випадку результатом роботи нейронної мережі є оцінка вектора зміщення в центрі об’єкта. Залежно від змісту сцени вам може знадобитися навіть контекстна інформація, щоб надійно ідентифікувати об’єкти, що цікавлять. Включення контекстної інформації знову призводить до великої кількості додаткових параметрів, що призводить до більш складних класифікаторів. Для вирішення цієї проблеми були розроблені методи мультипрогнозування з різними роздільними можливостями [33, 50, 51], які поєднують інформацію зі значень пікселів, розташованих на різних рівнях піраміди ознак [61], але зосереджених в одному місці. Це надає контекстну інформацію для класифікатора, але обходить комбінаторний вибух у кількості параметрів. Однак класифікатор повинен чітко вивчити зміну масштабу. Недоліком методів на основі піраміди є те, що вони охоплюють невелику частину можливого простору

масштабу, оскільки на кожному рівні піраміди роздільна здатність зменшується вдвічі. Особливим типом нейронної мережі, яка містить інформацію про масштаб безпосередньо в піраміді ознак, є так звана нейронна мережа вищого порядку [47, 48]. Вона використовує техніку під назвою «грубе кодування» для створення внутрішнього представлення. Однак нейронні мережі вищого порядку також повинні чітко вивчати зміни масштабу. Їх слід використовувати з обережністю, оскільки схеми грубого кодування можуть спричинити згладжування, оскільки зображення з високою роздільною здатністю не розмиваються, поки не буде обчислено наступний рівень.

Рідкісні ситуації, такі як неповна видимість об’єктів, або наявність кількох об’єктів на зображенні, обробленому класифікатором, в даний час не у фокусі наукової спільноти. Проте існує експериментальна архітектура, яка може розпізнавати кілька об’єктів на зображеннях одночасно [45].

Рекурентні нейронні мережі (зі зворотнім зв’язком [62]) можуть бути використані для розробки спеціальних підходів до розпізнавання об’єктів [27]. Додатковою цінністю архітектури рекурентної мережі є її пам'ять: поточний стан містить інформацію про минуле, що може додати цінне розуміння контексту. Наприклад, рекурентну мережу було використано для виявлення на зображеннях розливів нафти [27]. Принцип повторюваності вводить усереднення, що забезпечує більш надійну роботу.

До бінарних зображень можна застосувати декілька методів виявлення та класифікації об’єктів [44, 38, 52, 53]. Хоча бінаризація значно спрощує проблему розпізнавання, вона зазвичай знижує точність розпізнавання нейронної мережі.

- Розпізнавання об'єктів за ознаками

Було розроблено декілька нейромережевих підходів до розпізнавання об’єктів на основі ознак [30, 31, 33, 56, 63, 64], у тому числі: прямі нейронні мережі, нейронні мережі Хопфілда, нечіткі нейронні мережі та непараметричні нейронні

мережі. Самоорганізуючийся граф Кохонена іноді використовується для вибору ознак перед розпізнаванням об’єкта [56, 69], хоча його також можна використовувати для класифікації об’єктів.

Порівняно з методами на основі пікселів, існує менше типів нейронних архітектур, розроблених для розпізнавання об’єктів на основі ознак, тому більшість зусиль зосереджена на розробці та виборі найкращих функцій для завдання розпізнавання. Загальним для багатьох методів на основі ознак є те, що зміни обертання та масштабування вирішуються підбором ознак. Проте навіть невелика кількість шуму може вплинути на обчислювані функції та погіршити продуктивність розпізнавання.

Тому основне завдання класифікатора – відфільтрувати можливі шуми та викривлення в ознаках. Крім того, коли об’єкт, який потрібно виявити, є великими, і можливе розташування потребує ретельного пошуку. Інакше нейронний класифікатор буде містити занадто багато параметрів, і його буде важко узагальнити.

Для розпізнавання об’єктів ознаки зазвичай відображають локальні геометричні властивості:

- точки з високою кривизною на контурі виявленого об'єкта [31];
- банки фільтрів, включаючи вейвлети [69];
- проекція зображення на осі x і y [65];
- головні компоненти, отримані з зображень;
- відстані до траєкторій в просторі ознак, що описують об'єкти при всіх обертах, рухах або масштабах [66];
- дескриптори Фур'є із зображень [64];

Для розпізнавання об’єктів також розроблено багато методів на основі функцій:

- лінійно масштабований простір [68];
- піраміда Гаусса [33];
- піраміда Лапласа [70].

Крім того, позиції виявлених граней можуть служити ознаками для класифікатора [33].

Який набір ознак найбільш підходить для конкретного завдання розпізнавання, залежить від зміни положення об’єкта та фону, орієнтації на площині та масштабу. Знаючи ступені свободи, з якими має працювати метод, необхідно вибрати відповідний набір ознак.

1.2 Класичні методи розпізнавання облич

- Аналіз головних компонент

Аналіз головних компонентів, також відомий як метод Кархунена-Лава, є одним із найпопулярніших методів для вилучення ознак і стиснення даних. Для розпізнавання обличчя можна використовувати ознаки, вилучені за допомогою аналізу основних компонентів[71]. Існують також методи розпізнавання, які використовують власнимі вектори ознак, аби зменшити розмірність вихідного простору даних. Цей зменшений простір даних використовується для ідентифікації. Однак погана міжкласова роздільна здатність і висока обчислювальна складність є добре відомими і поширеними проблемами в цих методах. Це обмеження було подолано за допомогою лінійного дискримінантного аналізу (ЛДА). ЛДА є основним принципом видалення ознак у методах на основі зовнішнього вигляду. Але багато систем розпізнавання облич на основі ЛДА спочатку використовують аналіз головних компонентів, щоб зменшити розмір, а потім використовують ЛДА, щоб максимізувати роздільну здатність вилучених ознак. Причина в тому, що ЛДА

страждає від проблеми малого розміру вибірки, тобто вибраний набір даних повинен мати більші вибірки на клас, щоб надійно розрізняти описові ознаки. Таким чином, впровадження ЛДА безпосередньо призвело б до вилучення гірших ознак. Найбільш ефективним варіантом залучення ЛДА є використання його після того, як за допомогою фільтрів Габора визначається місцезнаходження фронтальних зображень облич, потім над ними проводиться аналіз головних компонентів для зменшення розмірності відфільтрованих векторів ознак, і лише в кінці ЛДА використовується для виділення ознак[73].

Рекурсивний алгоритм обчислення дескриптивних ознак процедури аналізу головних компонент та ЛДА введено в [74]. Метод зосереджений на складній проблемі обчислення описових векторів, тобто поетапних обчислень на потоках даних, які все більших розмірів, без обчислення відповідної коваріаційної матриці та без попереднього знання даних. Запропонований інкрементальний алгоритм дуже ефективний у використанні пам’яті та обчисленні перших базисних векторів. У порівнянні з дуже відомими алгоритмами розпізнавання облич, такими як методи аналізу головних компонентів і ЛДА, цей алгоритм забезпечує прийнятний рівень успіху в розпізнаванні обличчя.

У разі невеликих баз даних аналіз головних компонентів може перевершити більшість інших методів. Однак недоліком цього підходу є те, що він обчислювально складний по мірі збільшення розміру бази даних, оскільки всі пікселі в зображенні необхідні для отримання представлення для перевірки схожості вхідного зображення з усіма іншими зображеннями в базі даних.

Різні методи зменшення розмірності, такі як аналіз головних компонентів, ЛДА, локальна проекція та видалення ознак відносної відстані, були обрані та застосовані, щоб зменшити втрату ефективності класифікації через зміни зовнішнього вигляду обличчя. Ефективність розпізнавання при використанні аналізу головних компонент та ЛДА задля зменшення розмірності є приблизно однаковою з точки зору точності. Але було помічено, що ЛДА займає багато часу для обробки зображень кількох облич, навіть для невеликих баз даних. У випадку

проекції зі збереженням локальності, і вилучення ознак, що зберігають відносну відстань, коли кількість зображень обличчя стає більше, швидкість розпізнавання значно нижча порівняно з методом аналізу головних компонентів.

Існує удосконалений алгоритм аналізу головних компонентів для розпізнавання обличчя, який заснований на ідеї зменшення впливу власних векторів [75], пов’язаних з великими власними значеннями, шляхом нормалізації елементно-векторних ознак на його відповідне стандартне відхилення. Результати моделювання показують, що запропонований метод має кращу продуктивність, ніж традиційні методи основних компонент та ЛДА, а обчислювальна вартість така ж, як і простого аналізу головних компонентів, і значно нижча, ніж у ЛДА.

Найбільш просунутий метод розпізнавання облич, заснований на аналізі головних компонентів включає в себе ЛДА та нейронні мережі [76]. Метод включає чотири кроки:

1. Попередня обробка зображення.
2. Зменшення розміру шляхом аналізу основних компонентів.
3. Виділення ознак за допомогою ЛДА.
4. Класифікація за нейронною мережею.

Коли доступні кілька зразків зображень, аналіз головних компонентів і ЛДА вдало поєднують свої сильні сторони, і, використовуючи нейронний класифікатор, можна суттєво зменшити кількість помилок, спричинених нелінійним розділенням класів.

Також існує метод виявлення обличчя за допомогою мережі радіальних базисних функцій [77], який мінімізує час обчислень при досягненні більш високої точності. Аналіз головних компонент використовується для зменшення розміру власних векторів. Мережа радіальної базисної функції використовується для апроксимації функції, яка визначає, чи містить вона вхідне зображення обличчя, і якщо так, повідомляє його орієнтацію. Запропонована система показує, що мережі

радіальних базисних функцій можуть показувати кращі результати, ніж нейронні мережі, навчені за алгоритмом зворотного поширення.

- Метод опорних векторів

Метод опорних векторів є одним із найбільш корисних у задачах класифікації. Прикладом є розпізнавання обличчя. Однак цей метод не можна використовувати, якщо в базі даних векторів ознак відсутні записи. Перевага методу опорних векторів перед традиційними нейронними мережами полягає в тому, що метод опорних векторів зазвичай демонструє кращу здатність до узагальнення на малих вибірках даних.

- Аналіз незалежних компонент

Аналіз незалежних компонент є методом знаходження головних компонентів багатовимірної статистики. Аналіз незалежних компонент відрізняється від інших методів тим, що він шукає статистично незалежні компоненти, які не є гауссовими. Аналіз незалежних компонент зводиться до пошуку лінійного представлення, де компоненти статистично незалежні. Порівняння розпізнавання облич між аналізом головних компонентів та аналізом незалежних компонентів різних класифікаторів виявило, що незалежний компонентний аналіз мав кращі показники розпізнавання, ніж аналіз головних компонентів зі статистично незалежними базовими зображеннями [78]. У аналізі незалежних компонент кожне зображення обличчя перетворюється у вектор перед обчисленням незалежних компонентів. Аналіз незалежних компонентів зменшує похибку розпізнавання обличчя, а розмірність підпростору розпізнавання стає меншою. У зв'язку з цим новий метод розпізнавання обличчя, який поєднує незалежну модель аналізу компонентів з методом оптичної кореляції [79]. Цей підхід ґрунтується на ефективності методів оптичної кореляції з чіткими відмінностями та надійністю моделі незалежних компонентів. Моделі аналізу незалежних компонентів викликали інтерес до пошуку лінійних перетворень для вираження набору випадкових величин у формі

лінійних комбінацій статистично незалежних вихідних змінних [80]. Аналіз незалежних компонентів забезпечує більш чітке представлення даних, ніж аналіз основних компонентів, оскільки він має на меті забезпечити незалежну, а не некорельовану декомпозицію та представлення зображень.

1.3 Новітні методи та системи штучного інтелекту розпізнавання облич

- Вейвлет Габора

Для того, щоб покращити здатність розпізнавання обличчя, вектор ознак, витягнутий за допомогою вейвлет-перетворення Габора в зображенні обличчя, об'єднано з методом аналізу незалежного компонента в [81]. Орієнтири, видалені за допомогою вейвлетів Габора, вважаються одним із найкращих методів розпізнавання обличчя. В останні роки вейвлети Габора широко використовувалися дослідниками з розпізнавання облич для зображення облич, оскільки ядро вейвлетів Габора подібне до рецепторних полів ссавців. Вони демонструють бажані властивості просторової локалізації та селективності орієнтації. Попередня робота з видалення ознак за допомогою вейвлетів Габора також показала вражаючі результати розпізнавання обличчя. Знаки Габора також використовуються для розпізнавання ходи та статі в [82]. У роботі [83] було зазначено, що, хоча фази Габора чутливі до змін, вони можуть розрізняти моделі подібної величини, тобто надають більш детальну інформацію про зображення. Тому він працює відносно добре, якщо чутливість фази Габора до зміщень і локальних змін може бути компенсована. Також популярним у літературі є використання локальних бінарних шаблонів Габора для представлення зображень, які поєднують значення Габора з операторами локальних бінарних шаблонів [84]. Оскільки риси обличчя, отримані з локальних бінарних шаблонів Габора, засновані на локальних гістограмах, які нечутливі до локальних змін [85], локальні гістограми також можна використовувати для зниження чутливості фази Габора до

локальних змін. Кодування фази Габора за допомогою локальних двійкових шаблонів і локальних гістограм дозволяє досягти дуже вражаючої продуктивності розпізнавання. Запропоновано інший метод виділення рис обличчя на основі вейвлет-перетворення зображень обличчя та алгоритму найменших квадратів [86]. Результати експерименту показали, що метод дискримінації за методом найменших квадратів на основі вейвлету Габора перевершує методи вилучення ознак, такі як аналіз основних компонент, дискримінантний аналіз, а також комбінацію цих методів із зображеннями обличчя.

Існують також методи вилучення векторів ознак із вейвлет-перетворення зображень обличчя в поєднанні з лінійним дискримінантним аналізом, щоб підвищити точність розпізнавання обличчя порівняно з простим вейвлет-перетворенням [87].

Серед нових методів у літературі для розрізнення ознак було показано, що фільтри Габора вилучають найбільшу кількість інформації з локальних областей зображення і є інваріантними до змін, викликаних рухом, обертанням, освітленням і масштабом [88]. Комбінація вейвлетів Габора та нейронних мереж була запропонована для виявлення облич, і надано гібридне рішення нейронної мережі для розпізнавання обличчя, навчене на ознаках, витягнутих за допомогою вейвлетів Габора[89]. Також вейвлети Габора використовуються для створення представлення облич, на яких потім навчають нейронні мережі для класифікації облич [90]. Розмірність зменшенують за допомогою аналізу головних компонентів. Інший метод, розроблений для вилучення векторів ознак цілого обличчя в базу даних зображень за допомогою фільтра Габора, який, як відомо, інваріантний до освітлення та представлення облич. Коли розмірність вектора ознак невелика, мережа досягає вищого рівня розпізнавання та кращої ефективності класифікації[91].

- Дискримінантний аналіз

Дискримінантний аналіз є потужним методом розпізнавання обличчя. Він надає функції шляхом лінійного перетворення вихідного простору даних у простір об’єктів низької розмірності з добре розділеними даними. Метод аналізу підпростору для розпізнавання обличчя, який називається дискримінаційною проекцією ядра, що зберігає положення, використовує напрацювання, взяті з основ дискримінантного аналізу, проекцій, що зберігають локалькність, і методів ядра [92]. Нелінійні підпростори, синтезовані цими методами, можуть не тільки зберегти локальну структуру різноманіття облич, але й підкреслити описову інформацію.

У поєднанні з критерієм максимального відступу у [93] був запропонований новий метод, щоб знайти підпростір, який найкраще розрізняє різні обличчя і зберігає внутрішні зв’язки локального сусідства в межах одного класу. Запропонований метод було порівняно з аналізом головних компонент, і автори показали, що він забезпечує краще представлення інформації про клас і досяг кращої точності розпізнавання. Адаптивний дискримінантний аналіз освітлення був запропонований в [94] для вирішення проблем зміни освітлення в розпізнаванні обличчя. Точність розпізнавання запропонованого методу набагато вища, ніж у методу головних компонент та класичного дискримінантного аналізу.

- Штучні нейронні мережі

Існують два основних напрямки досліджень для навчання нейромереж для розпізнавання облич. Деякі навчають мультикласовий класифікатор, який може відокремлювати різних людей в навчальному наборі, а інші навчають безпосередньо ознаки, які вилучає з зображення нейромережа, наприклад за допомогою критерію триплету [95]. На основі величезної кількості навчальних даних і складної архітектури нейромереж, як методи на основі класифікації, так і на основі триплетів можуть отримати високу точність при розпізнаванні обличчя.

Спостерігаючи, що коефіцієнти ваги з останнього шару нейромережі, натренованого на задачі класифікації, мають концептуальну схожість із центрами кожного класу облич, у [96] було запропоновано при тренуванні додавати штраф

кутового запасу, щоб запровадити додаткову внутрішню компактність класів і міжкласову розбіжність одночасно, що призводить до кращої дискримінаційної сили навченої моделі.

Незважаючи на те, що SphereFace [96] представили важливу ідею кутового запасу, їхня функція втрат вимагала ряду наближень для обчислення, що призвело до нестабільного навчання мережі. CosFace [97] безпосередньо додає косинусний штраф до цільового логіта, що призводить до кращої збіжності порівняно зі SphereFace, але допускає набагато легшу реалізацію та стабілізує навчання, суттєво зменшуючи вплив погрішності введеної використанням чисел з плаваючою комою.

РОЗДІЛ 2

АНАЛІЗ АРХІТЕКТУРНИХ РІШЕНЬ І ВИБІР ПРОГРАМНИХ ЗАСОБІВ ДЛЯ СТВОРЕННЯ ВЕБ-СЕРВІСУ

2.1 Фреймворк Django для створення веб сайтів на Python

Перш ніж зрозуміти Django, давайте спочатку зрозуміємо, навіщо нам потрібен веб-фреймворк? Веб-фреймворк — це серверний фреймворк додатків, який призначений для підтримки розробки динамічних веб-сайтів. Фреймворк позбавляє розробників від багатьох клопот веб-розробки та її різних компонентів. Таким чином, вибір фреймоврку є дуже важливою частиною розробки веб-сервісів, оскільки він може значно полегшити або ускладнити життя розробника, оскільки їм не потрібно кодувати з нуля. На ринку доступні різні фреймворки для веб-розробки. Деякі з них перераховані нижче:

- Angular
- Django
- Express
- React JS
- Ruby on Rails

Однією з особливостей Django є те, що він побудований на Python. Він дотримується принципу «Не повторюйся». Як видно з назви, цей принцип полягає в тому, щоб код був простим і не повторювався. Він бере на себе аутентифікацію користувачів, адміністрування контенту сайту, карти сторінок, захист від можливих вразливостей коду та багато іншого.

Django дуже безпечний. Це допомагає розробникам уникнути багатьох поширених помилок безпеки, таких як:

- SQL-ін'єкція — це техніка ін’єкції коду, яка використовується для атаки додатків, керованих даними, у якій шкідливі оператори SQL вставляють у поле введення для виконання (наприклад, щоб надати вміст бази даних зловмиснику)
- Міжсайтовий скриптинг - може використовуватися зловмисниками для обходу контролю доступу

- Міжсайтова підробка запиту - це тип зловмисного використання веб-сайту, коли несанкціоновані команди надсилаються від користувача, якому веб-додаток довіряє
- клік-джекінг – це зловмисний прийом, який обманює користувача, щоб він натиснув на щось відмінне від того, що очікує користувач, таким чином потенційно розкриваючи конфіденційну інформацію або дозволяючи іншим отримати контроль над комп’ютером користувача

Його система аутентифікації користувачів забезпечує безпечний спосіб керування обліковими записами та паролями користувачів.

Він досить масштабований. Django дозволяє додавати обладнання на будь-якому рівні — сервери баз даних, сервери кешування або обчислювальні сервери.

Фреймворк чітко розділяє такі компоненти, як рівень бази даних і рівень програми. І він постачається з простою, але потужною структурою кешу, яка дозволяє ефективно обробляти потужні напливи трафіку. Кешувати щось означає зберегти результат дорогого обчислення, щоб потім не довелося виконувати це обчислення ще раз при новому запиті. Збереження і знаходження цього результату в кеші вимагає набагато менше часу і затрат електроенергії ніж повторне обчислення, тому хорошою практикою є використання кешу для всіх складних обчислень.

Django використовується для створення різноманітних речей (дуже універсальних) — від систем керування вмістом до соціальних мереж до наукових обчислювальних платформ. Він має ряд інших переваг, оскільки має автоматичний інтерфейс адміністрування, об'єктно-реляційне відображення, RSS-канали та багато іншого.

Архітектура Django

Django дотримується архітектури MVT або Model View Template, яка базується на архітектурі MVC або Model View Controller. Основна відмінність між

ними двома полягає в тому, що Django сам піклується про ту частину яку у MVC виконує контролер.

MVC використовується для розробки веб-додатків, де ми розбиваємо код на різні сегменти. В даному випадку 3 сегменти — модель, вигляд і контролер.

- Модель — використовується для зберігання, обробки та підтримки даних. Це бекенд, де зберігається база даних сайту і мають виконуватись усі важкі обчислення.
- Вигляд — у шаблонах Django вигляд представляється у форматі html. Вигляд пов'язаний лише з відображенням сторінок сайту користувачу і він зовсім не знає про те що відбувається на бекенді. Все, що бачить користувач, проходить через ці шаблони вигляду.
- Контролер — це частина яка виконує бізнес-логіку, вона взаємодіє з моделлю та виглядом, пересилаючи користувача зі сторінки на сторінку за заданим маршрутом у відповідь на дії користувача.

MVT означає шаблон вигляду і моделі (model view template). У MVT є попередньо визначений шаблон інтерфейсу користувача. Наприклад, якщо треба написати кілька статичних HTML-форм, таких як Добрий день Користувач 1, Добрий день Користувач 2, і так далі. З шаблоном для цього треба лише один файл, який виводить “Добрий день” разом із значенням змінної в якій зберігається ім’я користувача. Тепер ця змінна буде замінена в цьому конкретному шаблоні за допомогою формату шаблонів Jinja. Це головне призначення шаблону - зменшити переписування одного й того ж коду до мінімуму.

У випадку з MVT, Django сам піклується про частину яку виконує контролер. Тобто, можна сказати, це сам фреймворк: механізм, який надсилає запит до відповідного вигляду, відповідно до конфігурації URL-адреси Django.

Отже, у випадку Django кожен компонент виконує наступні функції:

- Модель допомагає працювати з базою даних. Це рівень доступу до даних, який містить необхідні дані і інструкції щодо їх обробки.. Модель — це клас Python, і вона нічого не знає про інші шари Django. Моделі допомагають розробникам створювати, читати, оновлювати та видаляти об’єкти у базі даних сайту. Крім того, вони містять інформацію про бізнес-логіку, специфічні для застосунку методи, налаштування та інші речі, пов’язані з маніпулюванням даними.
- Вигляд використовується для виконання бізнес-логіки та взаємодії з моделлю для донесення даних до користувача шляхом заповнення шаблону. Вигляд отримує дані з моделі. Потім він або надає кожному шаблону доступ до конкретних даних для відображення, або обробляє дані заздалегідь. Він приймає запити HTTP, застосовує бізнес-логіку, надану класами і методами Python, і надає відповіді HTTP на запити клієнта.
- Шаблон це рівень відображення, який повністю обробляє частину інтерфейсу користувача. Це файли з HTML-кодом, які використовуються для візуалізації даних для кінцевого користувача. Вміст цих файлів може бути статичним або динамічним. Шаблон використовується лише для представлення даних, оскільки в ньому немає бізнес-логіки.

«Вигляд» — це функція зворотного виклику Python для певної URL-адреси, оскільки ця функція зворотного виклику описує, які дані мають бути представлені користувачу. Це дуже зручно відокремлювати вміст від представлення — саме тут на допомогу приходять шаблони. У Django «Вигляд» описує, які дані представлені, але подання зазвичай визначає «Шаблон», який описує, як дані представлені.

«Контролером» тут є сам Django, який надсилає запит у відповідний «Вигляд» в залежності від вказаної URL-адреси. Ось чому Django називають архітектурою MTV, а не MVC. Розробник надає «Модель», «Вигляд» та «Шаблон», потім зіставляє їх із URL-адресами, а Django робить усе необхідне щоб подати їх користувачеві.

Ще однією перевагою Django є те, що він працює з більшістю основних форматів баз даних. Об’єктно-реляційне відображення Django сумісне з низкою популярних форматів баз даних, але його ключовою особливістю є те, що воно дозволяє розробникам працювати з кількома форматами баз даних одночасно. Більше того, Django дає змогу переходити з одного формату бази даних в іншу та виконувати операції без необхідності переписувати код взаємодії з базами даних.

Також цей фреймворк постійно розвивається завдяки своїй спільноті. Навколо Django є велика спільнота, до якої щодня приєднується все більше ентузіастів. Вони постійно оновлюють та покращують компоненти фреймворку, а також розробляють нові бібліотеки для вирішення проблем, з якими професіонали часто стикаються під час розробки веб-додатків. Обираючи Django для реалізації веб-сервісу ми маємо на меті полегшення майбутніх досліджень і покращень в області веб-застосунків для розпізнавання облич, оскільки фреймворк з часом буде лише покращуватись і все більше людей зможуть все простіше робити внесок до коду представленого в даній роботі.

2.2 Вибір програмної реалізації для модулю розпізнавання облич на сайті

Перш за все хочеться зазначити що перед тим як виконувати розпізнавання облич необхідним кроком є виявлення облич на фотографії. Виявлення облич — це важлива частина розпізнавання облич, яка визначає кількість облич на зображенні

чи відео без запам’ятовування чи збереження деталей. Він може визначати деякі загальні дані, наприклад вік чи стать, але не може розпізнати окремих осіб.

Розпізнавання обличчя ідентифікує обличчя на фотографії або кадрі відео за наявною базою даних облич. Для цього із зображення обличчя потрібно вилучити ознаки, які є достатньо дескриптивними для того щоб по ним можна було визначати схожість або відмінність облич. Після цього порівнює їх з інформацією, що зберігається в базі даних.

Зазвичай методи розпізнавання потребують подання зображення яке повністю заповнене обличчям інтересу. Тобто зображення має містити лише обличчя людини, без всього тіла і зайвого контексту на кшталт фону і особливо інших людей в кадрі. Саме тому для більшості систем розпізнавання моделей критичним моментом є те наскільки добре система може виявити усі обличчя людей на фотографії та вирізати кожне обличчя в окреме зображення.

Методи виявлення облич

Для того щоб виявити обличчя система спочатку розглядає фотографію або кадр відео і намагається відрізнити обличчя від будь-яких інших об’єктів на фоні. Існують методи, за допомогою яких комп’ютер може цього досягти, компенсуючи освітлення, орієнтацію або відстань камери. Ці методи поділяються на чотири категорії, і алгоритми розпізнавання обличчя можуть належати до декількох з цих категорій одночасно:

- *Розпізнавання обличчя на основі знань*

  Цей метод спирається на набір правил, розроблених людьми відповідно до наших знань. Ми знаємо, що обличчя повинно мати ніс, очі та рот на певній відстані та положенні один від одного. Проблема цього методу полягає в тому, що створити такий набір правил вкрай складно. Якщо правила занадто загальні або надто детальні, система

отримує багато помилкових спрацьовувань або замало правильних спрацьовувань відповідно. До того ж цей метод працює не для всіх кольорів шкіри і залежить від умов освітлення, які можуть змінити точний відтінок шкіри людини на знімку.

- *Відповідність шаблону*

  Метод відповідності шаблону використовує попередньо визначені або параметризовані шаблони облич, щоб знайти або виявити грані за кореляцією між попередньо визначеними шаблонами та вхідними зображеннями. Модель обличчя може бути побудована по основних краях обличчя за допомогою методу виявлення країв.

  Різновидом цього підходу є використання техніки контролю фону. Якщо є можливість отримати фронтальне зображення обличчя на простому фон, ви можете видалити фон, залишивши межі обличчя.

  Для цього підходу програмне забезпечення має кілька класифікаторів для виявлення різних типів фронтальних облич, а також деякі для профільних облич, таких як детектори очей, носа, рота, а в деяких випадках навіть всього тіла. Хоча цей підхід простий у застосуванні, він зазвичай недостатній для виявлення обличчя.

- *Розпізнавання обличчя на основі ознак*

  Метод на основі ознак виділяє структурні особливості обличчя. Він навчається як класифікатор, а потім використовується для того, щоб відрізняти лицьові та не лицьові області зображення. Одним із прикладів цього методу є розпізнавання обличчя на основі кольору, яке сканує кольорові зображення або відео на предмет ділянок із типовим кольором шкіри, а потім шукає сегменти обличчя.

  - Ознаки Хаара [98] спираються на подібні властивості людських облич, щоб знаходити обличчя знаходженням наступних рис обличчя: розташування та розмір ока, рота, перенісся та орієнтовані градієнти інтенсивності пікселів. Для отримання ознак Хаара використовується 38 шарів каскадних

класифікаторів, щоб отримати в загальній кількості 6061 ознаку з кожного фронтального фото обличчя.

- Гістограма орієнтованих градієнтів [99] — це ще один спосіб виділення ознак для виявлення об’єктів. Вилучені ознаки – це розподіл напрямків орієнтованих градієнтів зображення.

Градієнти, як правило, виявляють великі круглі края та кути, що дозволяють нам виявити області в яких ймовірно знаходяться окремі частини обличчя. Замість того, щоб розглядати інтенсивність пікселів, вони підраховують вектори градієнту, щоб локалізувати окремі сегменти зображення.

- *Розпізнавання обличчя на основі зовнішнього вигляду*

Більш просунутий метод, заснований на зовнішньому вигляді, залежить від набору тренінгових зображень обличчя делегатів для визначення моделей обличчя. Він покладається на машинне навчання та статистичний аналіз, щоб знайти відповідні характеристики зображень обличчя та виділити з них особливості.

Готові програмні рішення можна розділити на 2 типи: приватні та публічні.

Приватні рішення проблеми розпізнавання облич продаються як сервіс, де за кожне окреме розпізнавання обличчя треба платити кошти.

Публічні рішення характеризуються відкритим кодом і, зазвичай, публічно доступними параметрами моделей.

Перевагою приватних рішень є те, що турбуватися про безпечне утримання бази даних облич і вартість при обробці запиту а пошук обличчя в базі не потрібно. Вадою є те, що середня ціна запиту в такому випадку буде значно вищою ніж якщо брати всі ризики і задачі управління на себе. Також до недоліків можна віднести те, що такі сервіси можуть змінювати правила користування, ціну за користування в

односторонньому порядку. У випадку їх закриття для відновлення власного сервісу може знадобитися багато зусиль.

Також зачасту публічні рішення не можна використовувати в комерційних проектах, тому для таких задач може бути дешевше платити за сервіс який надає готові рішення ніж створювати власне рішення з нуля.

В ході роботи були розглянуті такі приватні рішення для веб-сервісу

- Amazon Rekognition заснований на глибокому навчанні і повністю інтегрований в екосистему Amazon Web Service. Це надійне рішення як для виявлення, так і для розпізнавання облич, і воно застосовне для виявлення восьми основних емоцій, таких як «щасливий», «сумний», «злий» тощо.
- Face++ — хмарний сервіс аналізу облич, який дозволяє виконувати необмежену кількість запитів, але не більше трьох за 1 секунду.
- Програмний інтерфейс для розпізнавання та виявлення облич від Lambda Labs надає можливості для розпізнавання обличчя, виявлення обличчя, положення очей, положення носа, положення рота та визначення статі. Він пропонує 1000 безкоштовних запитів на місяць.
- Kairos пропонує різноманітні рішення для розпізнавання. Їхні програмні інтерфейси надають можливість визначення статі, віку, розпізнавання обличчя та емоцій на фотографіях та відео. Вони пропонують 14-денну безкоштовну пробну версію з максимальним обмеженням у 10 000 запитів.
- Microsoft Azure Cognitive Services Face дозволяє робити до 30 000 запитів на місяць і до 20 запитів на хвилину на безкоштовній основі. Для платних запитів ціна залежить від кількості розпізнавань на місяць, починаючи з 1 доллару за 1000 розпізнань. Функціонал також включає оцінку віку, статі та розпізнавання емоцій.
- Paravision — це компанія яка займається розпізнаванням облич для підприємств і надає окремі рішення для кожного підприємства під їх

специфічні задачі. Серед їхніх відомих послуг — розпізнавання облич і виду активності та рішення щодо COVID-19 (розпізнавання облич з масками, інтеграція з термовиявленням тощо).

Публічні рішення для розпізнавання обличчя з відкритим кодом

- Ageitgey/face_recognition — це одна з найпоширеніших бібліотек для розпізнавання облич. Це один з найпростіших програмних інтерфейсів для розпізнавання обличчя на Python. Однак їх основний недолік — це відсутність регулярних оновлень, через що точність моделі залишає бажати кращого.
- Deepface — забезпечує аналіз атрибутів обличчя, таких як вік, стать, раса та емоції. Не підходить через те що атрибути хоча й можуть бути використані для розпізнавання обличчя, проте ознаки вивчені спеціалізованими на цьому моделі є набагато більш точними.
- FaceNet, розроблений Google, використовує бібліотеку Python для реалізації. Регулярні оновлення не проводяться.
- InsightFace — це репозиторій, який активно оновлюється. Точність розпізнавання становить 99,86%. Наразі це найбільш точний і найбільш добре оформлений проект з відкритим кодом.

За результатами дослідження різних готових рішень для модулю розпізнавання обличчя було вирішено зупинитися на рішені з відкритим кодом, аби результуючий проект було легко відтворити, а також аби сформувати повне розуміння того як працює ця система на всіх рівнях. Адже приватні рішення з метою запобігання конкуренції не розголошують які саме техніки використовуються на фоні коли сервіс робить запит на розпізнавання.

Серед усіх рішень з відкритим кодом однозначним фаворитом є репозиторій InsightFace. Він поєднує в собі усі можливі переваги: простий у використанні, має

регулярні оновлення, а також, найголовніше для сервісу з розпізнавання облич, - демонструє найбільшу точність розпізнавання.

2.3 Переваги та недоліки алгоритмів систем розпізнавання облич

Розпізнавання облич це технологія яка викликає багато дискусій на тему етичності її застосування і необхідності її подальшого розвитку. Ми маємо враховувати усі ризики які вона створює для нашого суспільства, але водночас варто пам’ятати про безперечні переваги і рішення багатьох проблем які вона пропонує.

Серед головних переваг технології можна назвати:

- Можна використовувати для авторизації
  Однією з переваг технології розпізнавання облич є те, що її можна використовувати для надання доступу до електронних пристроїв. Незважаючи на те, що технологія все ще не широко використовується, можливість розблокувати смартфон або комп’ютер, просто поглянувши на камеру, стає досить поширеною вже зараз.
- Збільшує рівень безпеки в соціумі
  Ще одна перевага розпізнавання облич полягає в тому, що воно також може допомогти підвищити загальний рівень безпеки в усьому світі. Злочинці двічі подумають, перш ніж займатися незаконною діяльністю, оскільки знають, що шанси бути спійманими значно збільшуються за наявності належної технології розпізнавання облич.
- Може допомогти контролювати поширення захворювань
  За допомогою розпізнавання облич ми також можемо уникнути поширення захворювань. Воно дозволяє складати списки усіх взаємодій з усіми громадянами що відбувалися у публічному просторі де є камери, тим самим сповіщаючи про ризики зараження у випадку контакту з інфікованими людьми. Також використання таких систем для авторизації у закладах зменшує необхідність безпосереднього контактування людей.

- Допомагає підвищувати ефективність процесів
  Багато закладів не потребують високого рівня захисту від посторонніх і тому доступ до них по розпізнаванню обличчя є цілком прийнятним. Наприклад, при користуванні громадським транспортом набагато зручніше щоб проїзд списувався з особистого рахунку користувача який прив'язаний до його обличчя, бо в такому випадку можна просто зайти в траспорт і одразу сісти на своє місце.

  До недоліків можна віднести:

- Проблеми з конфіденційністю
  Для того, щоб реалізувати систему, яка здатна ідентифікувати людські обличчя потрібно зібрати базу даних цих облич, що накладає великі ризики несанкціонованого доступу до них.
- Це може бути дорого
  Якщо така система впроваджена на державному рівні, це означає що необхідно підтримувати базу даних мільйонів облич, а також при кожному запиті порівнювати обличчя з мільйонами інших. На сучасному рівні розвитку технології це дуже дорого і далеко не всі платники податків на таке погодяться.
- Сприяє появі авторитарних режимів
  Держава може використовувати цю технологію для тотального контролю за життям людей. Для виявлення і ліквідації всіх членів усіх опозиційних рухів.
- Технологія ще недостатньо розвинена
  Не дивлячись на те, що значний прогрес вже було зроблено в плані точності сучасних систем розпізнавання обличчя, цього все ще не достатньо. Справа в масштабах застосування. Більшість досліджень проводиться на вибірках до сотен тисяч облич, в той час як для серйозного застосування на світовому рівні потрібно вміти точно визначити належність обличчя людині серед сотен мільйонів інших людей.

Загалом при правильному використанні дана технологія є надзвичайно корисною і перспективною, особливо у сфері безпеки.

РОЗДІЛ 3

ПРОЕКТУВАННЯ, РЕАЛІЗАЦІЯ І ВПРОВАДЖЕННЯ ПРОГРАМНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ОБЛИЧ

3.1 Постановка задачі

Основними компонентами необхідними для веб-застосунку з розпізнавання облич виділити такі основні компоненти як:

- База даних

У цій роботі центральним компонентом, який забезпечує виконання функціональності системи, є створення і використання бази даних. База даних облич являє собою таблицю яка містить в собі пари ознак - інформація про людину. Для того щоб створити можливість додавати рядок в цю базу даних необхідно вирішити наступні підзадачі:

  - зчитувати зображення, що надходять у стиснутому вигляді;
  - знаходження усіх облич, що містяться на оброблюванному зображені;
  - вирізати і масштабувати зображення з метою підвищення дескриптивності ознак, які зможе вилучити з зображення натренована модель;
  - вилучити ознаки з кожного окремого обличчя на зображенні;
  - зберігати вирізані і оброблені зображення облич у файловій системі комп’ютера для подальшого їх завантаження у випадку необхідності вивести фотографії доданої людини;
  - зберігати вилучені ознаки у файловій структурі, яка дозволить їх швидке зчитування з диску і запис в оперативну пам’ять, а також дозволить оновлювати модель і базу даних в асинхронному режимі. Тобто формат зберігання бази даних на жорсткому диску не має бути зосереджений на одному файлі, оскільки одночасно запис в файл не підтримується більшістю файлових систем.

- Можливість створення запитів для перевірки входження в базу даних нового зображення:
  - створити структуру для зберігання бази даних в оперативній пам’яті, яка дозволить проводити швидке та ефективне

порівняння схожості усіх ознак з бази даних з ознаками облич у запитах;

- зчитувати зображення, що надходять у стиснутому вигляді;
- знаходження усіх облич, що містяться на оброблюванному зображені;
- вирізати і масштабувати зображення з метою підвищення дескриптивності ознак, які зможе вилучити з зображення натренована модель;
- вилучити ознаки з кожного окремого обличчя на зображені;
- провести порівняння вилучених ознак з усіми ознаками у базі даних і на основі цього повертати найбільш схожих людей;

- Зрозуміле для користувача відображення результатів перевірки на входженя:
  - відображення коефіцієнту схожості обличчя, введеного користувачем з кожним із виданих облич із бази даних
  - виведення підказок щодо того, який коефіцієнт схожості відповідає тому, що знайдене обличчя:
    - імовірно тої ж самої людини;
    - можливо тої самої людини;
    - швидше за все не тої ж людини що на фото відправленому у запиті;
- Можливість додавання обличчя в базу даних
  Після проведення обробки обличчя користувач має мати змогу додати обличчя, яке він шукав до бази даних, прикріпивши до нього текстову інформацію, яка буде відображена у разі, якщо обличчя цієї людини буде знову додано у запит до бази даних.
- З’єднання бази даних і користувацького інтерфейсу
  Для того, щоб дозволити швидкий цикл розробки продукту, перезапуск всієї частини коду, що пов’язана з інтерфейсом має бути швидким і зручним. Проте, на ініціалізацію бази даних може йти багато часу, який

лінійно залежить від кількості записів у ній. Для того, щоб обійти цю проблему, варто відокремити процес, який підтримує базу даних. Цей окремий процес буде надавати програмний інтерфейс для частини проекту, яка стосується взаємодії користувача з базою даних.

3.2 Реалізація програмної системи розпізнавання облич

- Формування бази даних

Оскільки база даних є головним елементом всього проекту, то першим кроком реалізації системи стало формування бази даних облич.

Графічні дані, в тому числі зображення, за своєю природою є найбільш об’ємним типом даних. Тому їх зберігання і передача в нестиснутому вигляді взагалі не практикується. Для того щоб зчитувати стиснуті зображення потрібно використовувати декодер специфічний до формату стиснення. На додаток, в Python

є бібліотека OpenCV, яка не тільки дозволяє зручно працювати із зображеннями та відео, а й декодувати їх з усіх популярних форматів стиснення, тому проблема роботи зі стиснутими зображеннями вирішена саме за допомогою неї.

Знаходження усіх облич, що містяться на оброблюванному зображені це не тривіальна проблема, проте, InsightFace надає готове рішення для цієї проблеми. Для того щоб скористатися цим рішенням треба ініціалізувати їх модель в оперативній пам’яті та подавати зображення у форматі тривимірного масиву. Щоб зчитати зображення в такому форматі ми використовуємо функцію декодування з бібліотеки OpenCV і вона повертає зображення в потрібному форматі. Передавши зображення в модель надану InsightFace, ми отримуємо координати крайніх точок кожного обличчя знайденого на зображені. За замовчуванням у цій бібліотеці також відбувається обчислення віку, статі та більш детальних точок обличчя ніж потрібно для наших цілей, тому було прийнято рішення переписати цей метод залишивши лише вилучення необхідної інформації, що скорочує обчислювальні витрати в 3 рази, тобто суттєво економить час при обробці великих масивів даних, які будуть додаватися під час практичного використання системи.

Для того, щоб підвищити дескриптивну потужність ознак, які будуть вилучені за допомогою нейромережі InsightFace, центр обличчя (центром вважається ніс) має знаходитися у центрі зображення і повністю поміщатися у зображенні, проте при цьому займати якомога більшу площу зображення, тобто бути наближеним настільки, наскільки це можливо. На рис. 3.1 зображено приклади того, як обличчя зняті в різних ракурсаху у результаті виглядають після цих оброб.

Рис 3.1 - Приклади вирізаних зображень з відцентрованими обличчями

Після цього оброблені зображення облич масштабуються до розміру 114x114 пікселів та подаються на вхід до нейромережі, яка навчена для вилучення ознак, які дозволяють чітко розрізняти належність обличчя тій чи іншій людині.

У результаті роботи нейромережі отримуємо 512-вимірний вектор ознак, напрямок якого вказує на внутрішнє представлення заданого обличчя нейронною мережею. Ці вектори і є основною частиною сформованої бази даних. Для того, щоб виміряти схожість двох облич, то достатньо виміряти наскільки напрями їх відповідних векторів схожі. Для цього ми можемо скористатися формулою обчислення косинусу кута між векторами:

$$cos\,\alpha = \frac{\bar{a} \cdot \bar{b}}{|\bar{a}| \cdot |\bar{b}|}$$

таким чином, схожість двух облич за вказаною метрикою може бути у діапазоні [-1, 1], де -1 означає повністю протилежні обличчя, а 1 означає повністю ідентичні обличчя. Оскільки в багатовимірному просторі дуже багато ортогональних напрямків, то слід очікувати, що більшість векторів несхожих облич можуть мати косинус кута між один одним, близький до 0, і, при цьому, проблеми з вміщенням великої кількості попарно перпендикулярних векторів у нейромережі не буде. На рис 3.2 наведено спрощений приклад того, як результат підрахунку косинусів між векторами ознак облич можна використовувати для розпізнавання людей.

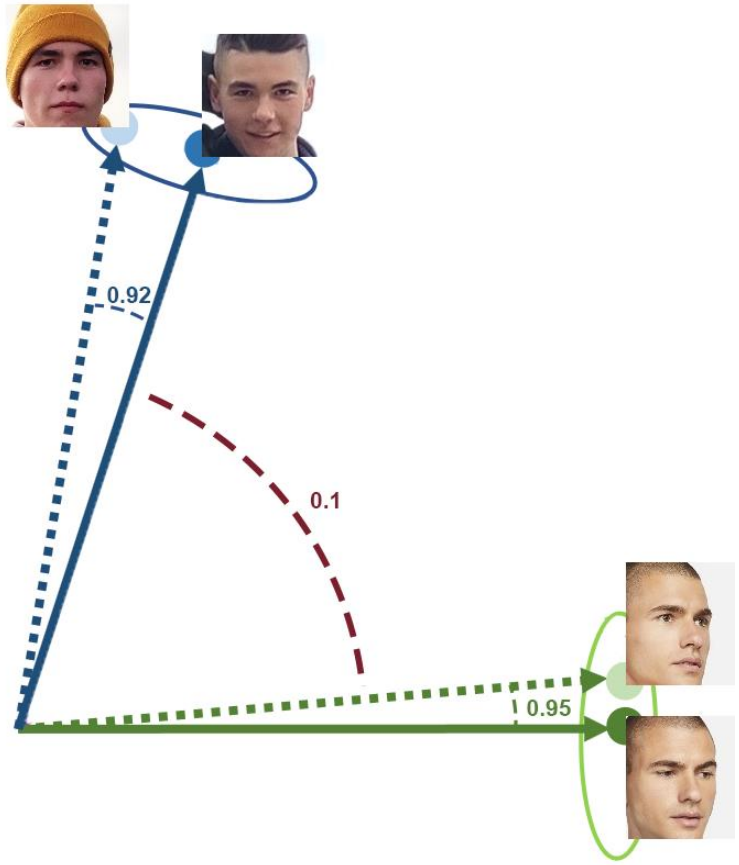

Рис 3.2 - Спрощений приклад підрахунку косинусів між векторами ознак облич

База даних зображень облич зберігається на диску у форматі довільного дерева папок, в яких зберігаються зображення стиснуті за допомогою Jpeg алгоритму, оскільки він надає найкращий компроміс між обсягом файлу та збереженням якості. Довільна ієрархія дозволяє гнучко та швидко маніпулювати базою даних у разі потреби - вилучати частину простою взаємодією з файловою системою без написання спеціально призначеної для цього програми.

Для того, щоб швидко зчитувати всю базу даних використовується кешування ознак. На старті сервісу для взаємодії з базою даних, він перевіряє існування файлу кешу для кожного зображення. Для зображень у яких є кешовані ознаки, ми просто зчитуємо ці ознаки з кешу і видаляємо ці зображення зі списку для зчитування. У випадку, якщо список зображень залишився не порожнім, проходимось по кожному зображенню в ньому, вилучаємо ознаки за допомогою методу описаного вище і зберігаємо їх у кеш для того, щоб наступного разу не робити ці обчислення знову.

- Можливість створення запитів для перевірки входження в базу даних нового зображення

Оскільки для обробки кожного запиту потрібно перевірити косинус куту між кожним вектором із наявних у базі даних і вхідним вектором, дану операцію можна представити як перемноження матриці розміром Nx512 на вектор розміром 512, де N - кількість записів у базі даних. Дане представлення не лише спрощує математичне представлення роботи бази даних, а й дозволяє виконувати порівняння значно швидше, адже сучасні бібліотеки для високопродуктивних обчислень особливу увагу звертають на оптимізацію операції перемноження матриць. Сучасні процесори мають інструкції, які дозволяють за один такт обчислити результат поелементного перемноження і сумації цілого рядка матриці, що прискорює виконання тих самих операцій в сотні разів порівняно з наївною реалізацією даних обчислень.

Під час ініціалізації, кожен вектор у базі даних слід нормалізувати - поділити на його довжину. Це зекономить дуже багато обчислень, аби не вираховувати довжину усіх векторів з бази даних кожного разу. Також не має сенсу ділити усі N значень, отриманих після перемноження матриці з вектором, на норму вхідного вектору, адже ця операція еквівалентна нормалізації вектору перед перемноженням, але вимагає набагато більше обчислень.

Для того щоб з отриманих N значень отримати M найбільш великих достатньо виконати часткове сортування лише на області останніх M чисел, що зменшує обчислювальну складність до O(N) порівняно зі звичайним сортуванням, яке виконується за O(NlogN).

Отримавши M найбільших косинусів, ми відстежуємо з якими векторами з бази даних ці косинуси були отримані і повертаємо інформацію про записи з бази даних які відповідають ідентифікаторам цих векторів. Якщо модель яка вилучає ознаки працює коректно то серед цих M записів очікується наявність записів, які стосуються людини, обличчя якої було у запиті, проте лише в тому випадку, якщо ця людина наявна в базі.

Обробка зображення для перетворення у вектор ознак під часобробки запиту є аналогічною до обробки зображення під час додавання облич в базу даних, оскільки для того щоб ознаки було коректно порівнювати, вони мають бути отримані однаковим шляхом.

- Зрозуміле для користувача відображення результатів перевірки на входженя

Для того щоб система визначення була більш прозорою і можливі помилки помічені користувачем, було вирішено не переносити всю відповідальність рішення про однозначне твердження, що на зображеннях виявлені обличчя однієї людини або декількох різних людей. Натомість ми виводимо косинус куту між векторами ознак зображень, який люди добре розуміють на інтуїтивному рівні як рівень схожості облич.

Аби ввести користувача в контекст того, які рівні схожості є достатніми, а які швидше за все свідчать про те, що на зображеннях різні люди вирішено відображати число, яке позначає схожість, відповідним кольором. Якщо зображення імовірно містять обличчя однакової людини, то колір буде зеленим, якщо ж імовірно різних людей, то колір червоний. У випадку коли система не впевнена колір буде ближче до жовтого. Навіть якщо референсні будуть неточними, користувачі дуже швидко підсвідомо звикнуть і вивчать закономірність які відтінки означають яку ймовірність що перед ними та сама людина що вони відправили у запиті.

- Можливість додавання облича в базу даних

Для того щоб база даних застосунку була більш динамічною і дозволяла кращу маштабованість з точки зору наповнення даними, необхідним елементом інтерфейсу є можливість додавання нового обличчя за допомогою користувацького інтерфейсу. Для того щоб це реалізувати було вирішено на сторінці видачі результатів робити обличчя доступними для натискання. У разі натиснення на них, відкривається наявна текстова інформація про зображення, яку при цьому можна відредагувати.

При додаванні нового обличчя в базу даних є необхідність зміни розміру матриці ознак з Nx512 до (N+1)x512. Кожного разу робити таку операцію занадто дорого, тому можна використовувати інший підхід: під час ініціалізації ми виділяємо рівно N рядків пам’яті для векторів ознак. У випадку якщо треба розширити матрицю ще одним рядком, замість того щоб реініціалізувати матрицю з одним новим рядком, створюється матриця з двократним розміром (2 * N) цих рядків і зберігається вказівних на реальну кількість заповнених рядків, тобто в даному випадку N + 1. В такому разі, для того щоб додати ще один елемент, потрібно просто записати в рядок на який вказує вказівник новий вектор ознак і обчислювально складна операція створення нового масиву не виконується аж поки додавання нового рядка не буде виконано N раз з моменту реініціалізації.

- З’єднання бази даних і користувацького інтерфейсу

Програмний інтерфейс бази даних було вирішено зробити на Flask. Цей мережевий фреймворк дозволює багатопоточно оброблювати HTTP запити у стійкому до помилок стилі. В ньому не має ніяких зайвих функцій, що робить його простим для розуміння і швидким.

Принцип роботи програмного інтерфейсу наступний: на старті сервісу бази даних, коли закінчилась загрузка усіх ознак з жорсткого диску в оперативну пам’ять, запускається сервер Flask. В нього є один обробник подій, який реагує на POST-запит, який приходить по адресу ip:port/api/ai і очікує, що тіло запиту буде мати наступну структуру:

```
ai{
  img: bytearray;
}
```

Очікується, що масив байтів, який приходить в полі ‘img’, це закодоване в форматі JPEG зображення. У випадку, якщо декодування JPEG декодером не вдається, у відповідь на запит повертається опис помилки і код результату 500.

В іншому випадку, з зображення за процесом описаним вище вилучаються ознаки усіх наявних облич, і для кожного з них повертається вирізане форматоване зображення самого обличчя запиту, а також ряд із K зображень і описів найбільш схожих облич.

3.3 Тестування і впровадження системи розпізнавання облич

Для того, аби тестувати систему у масштабі, потрібно зібрати початкову базу даних. Було вирішено зробити це автоматично за допомогою написання програми, яка б ходила по сторінках веб-сайтів з обличчями і зберігала їх на диск, тобто займалась Web Crawling'ом.

Реалізація даної програми була реалізована на основі бібліотеки Selenium, яка дозволяє використовувати усі популярні веб-браузери в автоматичному режимі за допомогою простого програмного інтерфейсу з різних мов програмування, серед яких є і Python.

Selenium надає такі інструменти для збору даних з мережі інтернет:

- дозволяє автоматично переходити на вказані веб-адреси;
- взаємодіяти з кнопками і текстовими полями на сайтах;
- переходити зі сторінки на сторінку натискаючи внутрішні посилання;
- переключатися між різними вкладками як звичайний користувач;
- шукати елементи сайту по дереву HTML-синтаксису за допомогою ряду різних мов запитів, в тому числі XPath;
- маючи інтерфейс, виконаний в стилі Python, дана бібліотека легко інтегрується в усю логіку реалізовану на ньому в попередніх пунктах;

Збір даних налаштований під сайти-каталоги і відбувається за наступним алгоритмом:

- Програма заходить на сайт інтересу.
- За допомогою виконання запиту XPath отримує список усіх елементів каталогу.

- В циклі проходить по кожному елементу, знайденому за запитом і виконує таке:
    - За допомогою команди Selenium гортає сторінку до активного елементу, імітуючи звичайного користувача;
    - Емулює натискання клавіші контрол і кліку по елементу, аби відкрити посилання у новій вкладці;
    - Виконує запит XPath, який знаходить усі зображення на сторінці;
    - За допомогою вбудованого модулю urllib, відправляє запит на отримання зображення за вказаними у тегі зображень посиланнями;
    - Декодує отриманий результат з JPEG та пропускає через мережу, яка знаходить усі обличчя на фото;
    - Вирізає та записує всі частини зображень, які містять обличчя, в окремі файли на диску;
    - Закриває активну вкладку, повертаючись у вкладку, яка знаходиться на сторінці зі списком;
- Намагається прогортати вниз на 1000 пікселів, у випадку якщо це вдається, повертається до попереднього пункту, оскільки це значить, що після прогортування завантажились нові елементи. Якщо нових елементів після прогортування не з’явилося, програма завершується.

Для додавання вилучених тестових зображень у базу даних ознак на сайті, завдяки архітектурі зчитування бази даних з диску, усе що треба зробити - це перенести папки із вилученими зображеннями в ієрархію папки збереження бази даних.

Основною проблемою яка вимагає найбільш ретельного вивчення є швидкодія системи, тому тестування перш за все включає оцінку складності ініціалізації бази даних та оцінку складності обробку запиту.

В результаті автоматичного збору даних було зібрано 11671 зображень людських облич, що є недостатнім для створення корисної системи розпізнавання облич, проте достатньо для тестових цілей.

Оскільки усі операції мають лінійну складність від кількості записів в базі, то середній час на виконання операції наведений у таблиці 3.1 дозволяє оцінити потенціал для масштабування наведеної системи.

Таблиця 3.1 - Результати вимірювань швидкодії

Операція	Час (с)	Час / 1000 зображень у базі даних (с)
Ініціалізація бази даних без кешу	600	51.4
Ініціалізація бази даних з кешу	16	1.37
Перевірка на входження	0.19	0.016

Варто зазначити, що оцінка є песимістичною, оскільки окрім алгоритму з лінійною складністю на час обробки запиту також впливає швидкість зчитування і декодування найбільш схожих зображень після видачі. Ці операції мають сталий час, який не залежить від кількості записів у базі даних, а отже з додаванням нових даних вплив цих операцій буде наближатись до нуля.

Також важливим аспектом є вимірювання пам’яті, необхідної для збереження бази даних. Основне місце займають зображення облич та ознаки які з них були вилучені. У таблиці 3.2 наведено виміри займаного об’єму

Таблиця 3.2 - Обсяг бази даних на диску

Тип даних	Об'єм на диску (мегабайт)	Об'єм / 1000 зображень (мегабайт)
Зображення	64.8	5.51
Ознаки	24.2	2.06

Для підтримування і оновлення функціоналу бази даних важливо мати автоматичні тести, які перевіряють працездатність основних її інтерфейсів. Для цього використовуються такі прості перевірки базової логіки як:

- Перевірка на додавання

  В ході тестування данного інтерфейсу ми відправляємо в менеджер бази даних запит про додавання нового запису. У разі успішного виконання, розмір бази даних має збільшитись на 1.
- Перевірка пошуку по базі даних

  Для того, аби перевірити чи працює пошук по базі даних, після заповнення бази випадковими ознаками ми додаємо новий рядок випадкових ознак і одразу ж відправляємо запит на пошук цих ознак в базі даних. Очікується що співпадіння буде мати значення схожості по косинусу більше ніж 0.99, оскільки точне значення 1 зачасту буде не виконане через неточність обчислень чисел з плаваючою комою.
- Перевірка видалення

  Перевірка видалення елементу відбувається з використанням тих ознак які ми додали на попередньому кроці. В цей раз вони видаляються з бази даних і після цього знову йде перевірка пошуку по базі даних, але в цей раз очікується що ця перевірка не буде пройдена, тобто база даних не має знайти запис після того як він був видалений.

ВИСНОВОК

У процесі написання цієї роботи було виконано такі завдання:

- Досліджено теоретичні основи аналізу зображень облич на основі нейромереж.
- Вивчено основи теорії штучного інтелекту при аналізі зображень.
- Визначено відомі методи та системи штучного інтелекту розпізнавання облич.
- Здійснено аналіз програмно-технічних рішень розпізнавання облич.
- Розглянуто фреймворки для створення веб сайтів на Python.
- Використано напрацювання з проекту з відкритим кодом InsightFace.
- Проаналізовано переваги та недоліки алгоритмів систем розпізнавання облич.
- Спроектовано, реалізовано і впроваджено програмну систему розпізнавання облич.

За результатами проведеної роботи було створено готовий до використання прототип веб-застосунок розпізнавання облич.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. J. Cornfield, Statistical classification methods, Proc. 2nd Conference on the diagnostic process, computer diagnosis and diagnostic methods, Chicago, 1972, pp. 108-130.

2. P.A. Devijver, J. Kittler, Pattern recognition: a statistical approach, Englewood Cliffs, London, 1982.

3. K. Fukunaga, Introduction to statistical pattern recognition, 2nd ed., Academic Press, New York, 1990.

4. D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: vol. I, Parallel Distributed Processing: Explorations in the microstructure of Cognition, D.E. Rumelhart and J.L. McClelland, eds., 1986, MIT Press, Cambridge, pp. 319-362.

5. J. Lampinen, E. Oja, Pattern recognition, in: Neural network systems, techniques and applications, vol. 5, Image processing and pattern recognition, C.T. Leondes, ed., 1998, Academic Press, pp. 1-59.

6. E. Oja, A simplified neuron model as a principal component analyzer, Journal of Mathematical Biology 15 (3) (1982) 267-273.

7. E. Oja, Neural networks, principal components, and subspaces, International Journal of Neural Systems 1 (1) (1989) 61-68.

8. P. Baldi, J. Hornik, Neural networks and principal component analysis: learning from examples without local minima, Neural Networks 2 (1) (1989) 53-58.

9. M. Kramer, Nonlinear principal component analysis using autoassociative neural networks, American Institute of Chemical Engineers Journal 37 (2) (1991) 223-243.

10. S. Usui, S. Nakauchi, M. Nakano, Internal color representation acquired by a five-layer neural network, Proc. International Conference on Artificial Neural Networks, Helsinki, Finland, 1991, pp. 867-872.

11. T. Hastie, W. Stuetzle, Principal curves, Journal of the American Statistical Association 84 (406) (1989) 502-516.

12. G.E. Hinton, P. Dayan, M. Revow, Modelling the manifolds of images of handwritten digits, IEEE Transactions on Neural Networks 8 (1) (1997) 65-74.

13. V.Z. Këpuska, S.O. Mason, A hierarchical neural network system for signalized point recognition in aerial photographs, Photogrammetric Engineering & Remote Sensing 61 (7) (1995) 917-925.

14. A. Shustorovich, A subspace projection approach to feature extraction - the 2-D Gabor transform for character recognition, Neural Networks 7 (8) (1994) 1295-1301.

15. P.N. Suganthan, H. Yan, Recognition of handprinted Chinese characters by constrained graph matching, Image and Vision Computing 16 (3) (1998) 191-201.

16. D. Patel, E.R. Davies, I. Hannah, The use of convolution operators for detecting contaminants in food images, Pattern Recognition 29 (6) (1996) 1019-1029.

17. J.O. Glass, W.E. Reddick, Hybrid artificial neural network segmentation and classification of dynamic contrast-enhanced MR imaging (DEMRI) of osteosarcoma, Magnetic Resonance Imaging 16 (9) (1998) 1075-1083.

18. R.J.T. Morris, L.D. Rubin, H. Tirri, Neural network techniques for object orientation detection: solution by optimal feedforward network and learning vector quantization approaches, IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (11) (1990) 1107-1115.

19. M. Fukumi, S. Omatu, Y. Nishikawa, Rotationinvariant neural pattern recognition system estimating a rotation angle, IEEE Transactions on Neural Networks 8 (3) (1997) 568-581.

20. C.K.I. Williams, M. Revow, G.E. Hinton, Instantiating deformable models with a neural net, Computer Vision and Image Understanding 68 (1) (1997) 120-126.

21. J. Lampinen, E. Oja, Distortion tolerant pattern recognition based on self-organizing feature extraction, IEEE Transactions on Neural Networks 6 (3) (1995) 539-547.

22. J.M. Cruz, G. Pajares, J. Aranda et al., Stereo matching technique based on the perceptron criterion function, Pattern Recognition Letters 16 (9) (1995) 933-944.

23. H.M. Abbas, M.M. Fahmy, Neural networks for maximum likelihood clustering, Signal Processing 36 (1) (1994) 111-126.

24. F.Y. Shih, J. Moh, F.-C. Chang, A new ARTbased neural architecture for pattern classification and image enhancement without prior knowledge, Pattern Recognition 25 (5) (1992) 533-542.

25. A.M. Waxman, M.C. Seibert, A. Gove et al., Neural processing of targets in visible multispectral IR and SAR imagery, Neural Networks 8 (7-8) (1995) 1029-1051.

26. C. Alippi, Real-time analysis of ships in radar images with neural networks, Pattern Recognition 28 (12) (1995) 1899-1913.

27. T. Ziemke, Radar image segmentation using recurrent artificial neural networks, Pattern Recognition Letters 17 (4) (1996) 319-334.

28. M. Egmont-Petersen, T. Arts, Recognition of radiopaque markers in X-ray images using a neural network as nonlinear filter, Pattern Recognition Letters 20 (5) (1999) 521-533.

29. M. Fukumi, S. Omatu, F. Takeda et al., Rotation-invariant neural pattern-recognition system with application to coin recognition, IEEE Transactions on Neural Networks 3 (2) (1992) 272-279.

30. R.E. Hurst, R.B. Bonner, K. Ashenayi et al., Neural net-based identification of cells expressing the p300 tumor-related antigen using fluorescence image analysis, Cytometry 27 (1) (1997) 36-42.

31. M.G. Penedo, M.J. Carreira, A. Mosquera et al., Computer-aided diagnosis: A neural-networkbased approach to lung nodule detection, IEEE Transactions on Medical Imaging 17 (6) (1998) 872-880.

32. L.I. Perlovsky, W.H. Schoendorf, B.J. Burdick et al., Model-based neural network for target detection in SAR images, IEEE Transactions on Image Processing 6 (1) (1997) 203-216.

33. M. Turner, J. Austin, N.M. Allinson et al., Chromosome location and feature extraction using neural networks, Image and Vision Computing 11 (4) (1993) 235-239.

34. L.C. Wang, S.Z. Der, N.M. Nasrabadi, Automatic target recognition using a featuredecomposition and data-decomposition modular neural network, IEEE Transactions on Image Processing 7 (8) (1998) 1113-1121.

35. A. Delopoulos, A. Tirakis, S. Kollias, Invariant image classification using triple-correlationbased neural networks, IEEE Transactions on Neural Networks 5 (3) (1994) 392-408.

36. Y. LeCun, L.D. Jackel, B. Boser et al., Handwritten digit recognition - applications of neural network chips and automatic learning, IEEE Communications Magazine 27 (11) (1989) 41- 46.

37. J.S. Lin, S.C.B. Lo, A. Hasegawa et al., Reduction of false positives in lung nodule detection using a two-level neural classification, IEEE Transactions on Medical Imaging 15 (2) (1996) 206-217.

38. G.A. Carpenter, S. Grossberg, G.W. Lesher, The what-and-where filter - a spatial mapping neural network for object recognition and image understanding, Computer Vision and Image Understanding 69 (1) (1998) 1-22.

39. L. Wang, S. Der, N. Nasrabadi, Composite classifiers for automatic target recognition, Optical Engineering 37 (3) (1998) 858-868.

40. N.K. Kasabov, S.I. Israel, B.J. Woodford, Adaptive, evolving, hybrid connectionist systems for image pattern recognition, in: Soft computing for image processing, S.K. Pal, A. Ghosh, and M.K. Kundu, eds., 2000, PhysicaVerlag, Heidelberg.

41. B. Kosko, Adaptive bidirectional associative memories, Applied Optics 26 (23) (1987) 4947- 4960.

42. K. Fukushima, Neocognitron: a hierarchical neural network capable of visual pattern recognition, Neural Networks 1 (2) (1988) 119-130.

43. C. Neubauer, Evaluation of convolutional neural networks for visual recognition, IEEE Transactions on Neural Networks 9 (4) (1998) 685-696.

44. J. Basak, S.K. Pal, PsyCOP - A psychologically motivated connectionist system for object perception, IEEE Transactions on Neural Networks 6 (6) (1995) 1337-1354.

45. M.R.J. McQuoid, Neural ensembles: Simultaneous recognition of multiple 2-D visual objects, Neural Networks 6 (7) (1993) 970-917.

46. J. Sklansky, M. Vriesenga, Genetic selection and neural modelling of piecewise-linear classifiers, International Journal of Pattern Recognition and Artificial Intelligence 10 (5) (1996) 587-612.

47. L. Spirkovska, M.B. Reid, Coarse-coded higherorder neural networks for PRSI object recognition, IEEE Transactions on Neural Networks 4 (2) (1993) 276-283.

48. L. Spirkovska, M.B. Reid, Higher-order neural networks applied to 2D and 3D object recognition, Machine Learning 15 (2) (1994) 169-199.

49. M. Antonucci, B. Tirozzi, N.D. Yarunin et al., Numerical simulation of neural networks with translation and rotation invatiant pattern recognition, International Journal of Modern Physics B 8 (11-12) (1994) 1529-1541.

50. S.S. Young, P.D. Scott, C. Bandera, Foveal automatic target recognition using a multiresolution neural network, IEEE Transactions on Image Processing 7 (8) (1998) 1122-1135.

51. S.S. Young, P.D. Scott, N.M. Nasrabadi, Object recognition using multilayer Hopfield neural network, IEEE Transactions on Image Processing 6 (3) (1997) 357-372.

52. S.S. Christensen, A.W. Andersen, T.M. Jørgensen et al., Visual guidance of a pig evisceration robot using neural networks, Pattern Recognition Letters 17 (4) (1996) 345-355.

53. E. do Valle Simões, L.F. Uebel, D.A.C. Barone, Hardware implementation of RAM neural networks, Pattern Recognition Letters 17 (4) (1996) 421-429.

54. B. Javidi, Q. Tang, Optical implementation of neural networks by the use of nonlinear joint transform correlators, Applied Optics 34 (20) (1995) 3950-3962.

55. J.-Y. Shen, Y.-X. Zhang, G.-G. Mu, Optical pattern recognition system based on a winnertake-all model of a neural network, Optical Engineering 32 (5) (1992) 1053-1056.

56. Y. Zheng, J.F. Greenleaf, J.J. Gisvold, Reduction of breast biopsies with a modified selforganizing map, IEEE Transactions on Neural Networks 8 (6) (1997) 1386-1396.

57. H.-P. Chan, S.-C.B. Lo, B. Sahiner et al., Computer-aided detection of mammographic microcalcifications: Pattern recognition with an artificial neural network, Medical Physics 22 (10) (1995) 1555-1567.

58. Y. LeCun, B. Boser, J.S. Denker et al., Backpropagation applied to handwritten zip code recognition, Neural Computation 1 (4) (1989) 541-551.

59. Z.G. Zhu, S.Q. Yang, G.Y. Xu et al., Fast road classification and orientation estimation using omni-view images and neural networks, IEEE Transactions on Image Processing 7 (8) (1998) 1182-1197.

60. Y. LeCun, B. Boser, J.S. Denker et al., Backpropagation applied to handwritten zip code recognition, Neural Computation 1 (4) (1989) 541-551.

61. B. Jähne, Digital image processing. Concepts, algorithms and scientific applications, Springer Verlag, Berlin, 1995.

62. Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Transactions on Neural Networks 5 (2) (1994) 157-166.

63. R. Anand, K. Mehrotra, C.K. Mohan et al., Analyzing images containing multiple sparse patterns with neural networks, Pattern Recognition 26 (11) (1993) 1717-1724.

64. J.Y. Wang, F.S. Cohen, 3-D Object recognition and shape estimation from image contours using b-splines, shape invariant matching, and neural network.2, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (1) (1994) 13-23.

65. R. Bajaj, Chaudhury, S., Signature verification using multiple neural classifiers, Pattern Recognition 30 (1) (1997) 1-7.

66. S. Deschênes, Y. Sheng, P.C. Chevrette, Threedimensional object recognition from twodimensional images using wavelet transforms and neural networks, Optical Engineering 37 (3) (1998) 763-770.

67. K. Huang, H. Yan, Off-line signature verification based on geometric feature extraction and neural network classification, Pattern Recognition 30 (1) (1997) 9-17.

68. H. Kai, H. Yan, Off-line signature verification based on geometric feature extraction and neural network classification, Pattern Recognition 30 (1) (1997) 9-17.

69. K.M. Iftekharuddin, T.D. Schechinger, K. Jemili et al., Feature-based neural wavelet optical character recognition system, Optical Engineering 34 (11) (1995) 3193-3199.

70. P. Sajda, C.D. Spence, S. Hsu et al., Integrating neural networks with image pyramids to learn target context, Neural Networks 8 (7-8) (1995) 1143-1152.

71. M. Turk and A. Pentland, "Eigenfaces for recognition," J. Cognitive Neuroscience,vol. 3, 71-86., 1991.

72. D. L. Swets and J. J. Weng, "Using discriminant eigenfeatures for image retrieval", IEEE Trans. PAMI., vol. 18, No. 8, 831-836, 1996.

73. C.Magesh Kumar, R.Thiyagarajan , S.P.Natarajan, S.Arulselvi,G.Sainarayanan,‖ Gabor features and LDA based Face Recognition with ANN classifier‖,Procedings Of ICETECT 2011

74. Issam Dagher,‖Incremental PCA-LDA algorithm‖, International Journal of Biometrics and Bioinformatics (IJBB), Volume (4): Issue (2)

75. Lin Luo, M.N.S. Swamy, Eugene I. Plotkin, ―A Modified PCA algorithm for face recognition‖, Proceedings of IEEE 2003

76. A. Hossein Sahoolizadeh, B. Zargham Heidari, and C. Hamid Dehghani,‖ A New Face Recognition Method using PCA, LDA and Neural Network‖, International Journal of Electrical and Electronics Engineering 2:8 2008

77. Feroz Ahmed Siddiky, Mohammed Shamsul Alam,Tanveer Ahsan and Mohammed Saifur Rahim,‖An Efficient Approach to Rotation Invariant Face detection using PCA,Generalized Regression Neural network and Mahalanobis Distance by reducing Search space‖,Proceedings Of IEEE 2007

78. Bruce A. Draper, Kyungim Baek, Marian Stewart Bartlett, ―Recognizing faces with PCA and ICA‖, Computer Vision and Image Understanding 91 (2003) 115-137.

79. A. Alfalou and C. Brosseau,‖ A New Robust and Discriminating Method for Face Recognition Based on Correlation Technique and Independent Component Analysis Model‖, Optics Letters 36 (2011) 645-647

80. P. Comon, ―Independent component analysis: a new concept?‖, Signal Process. 3, 287-314 (1994).

81. Arindam Kar, Debotosh Bhattacharjee, Dipak Kumar Basu, Mita Nasipuri, Mahantapas Kundu,‖ High Performance Human Face Recognition using Independent High Intensity Gabor Wavelet Responses: A Statistical Approach‖, International Journal of Computer Science & Emerging Technologies 178 Volume 2, Issue 1, February 2011

82. Li X, Maybank SJ, Yan S, Tao D, Xu D,‖Gait components and their application to gender recognition‖. IEEE Trans SMC-C 38(2):145–155,2008

83. Wenchao Zhang , Shiguang Shan,Laiyun Qing,Xilin Chen, Wen Gao,‖ Are Gabor phases really useless for face recognition?‖, Springer-Verlag London Limited 2008

84. Ahonen T, Hadid A, Pietika¨inen M,‖Face recognition with local binary patterns‖. In Proceeding of European conference on computer vision (ECCV2004), LNCS 3021, pp 469–481,2004

85. Hadjidemetriou E, Grossberg MD, Nayar SK,‖Multiresolution histograms and their use for recognition‖. IEEE Trans PAMI 26(7):831–84,2004

86. Vitomir ŠTRUC, Nikola PAVEŠI´C,‖ Gabor-Based Kernel Partial-Least-Squares Discrimination Features for Face Recognition‖,Informatica, , Vol. 20, No. 1, 115– 138,2009

87. Arindam Kar, Debotosh Bhattacharjee, Dipak Kumar Basu, Mita Nasipuri, Mahantapas Kundu,‖ High Performance Human Face ecognition using Independent High Intensity Gabor Wavelet Responses: A Statistical Approach‖, International Journal of Computer Science & Emerging Technologies (E-ISSN: 2044-6004) 178 Volume 2, Issue 1, February 2011

88. Z. Y. Mei, Z. Ming, and G. YuCong. ―Face recognition based on low dimensional gabor feature using direct fractional-step lda‖, In Proceedings of the Computer Graphics, Image and Vision: New Trends, IEEE Computer Society, 2005

89. A.Khatun and Md.Al-Amin Bhuiyan, ―Neural Network based Face Recognition with Gabor Filters‖, IJCSNS International Journal of Computer Science and Network Security, vol.11 No.1, January 2011.

90. P.Latha, L.Ganesan, N.Ramaraj, ―Gabor and Neural based Face Recognition‖, International Journal of Recent Trends in Engineering, Vol 2, No. 3, November 2009.

91. Anissa Bouzalmat ,Naouar Belghini ,Arsalane Zarghili,Jamal Kharroubi,Aicha Majda,‖ Face Recognition Using Neural Network Based Fourier Gabor Filters & Random Projection‖, International Journal of Computer Science and Security (IJCSS), Volume (5) : Issue (3), 2011

92. Rongbing Huang , Changming Su a, Fangnian Lang a, Minghui Du,‖ Kernel Discriminant Locality Preserving Projections for Human Face Recognition‖, Journal of Information & Computational Science 7: 4 (2010)

93. Xiaohu Ma¤, Yanqi Tan, Yaying Zhao, Hongbo Tian,‖ Face Recognition Based on Maximizing Margin and Discriminant Locality Preserving Projection‖, Journal of Information & Computational Science 7: 7 (2010)

94. Zhonghua Liu , Jingbo Zhou,Zhong Jin,‖ Face recognition based on illumination adaptive LDA‖, International Conference on Pattern Recognition,2010.

95. Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In CVPR, 2015.

96. Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In CVPR, 2017.

97. Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Zhifeng Li, Dihong Gong, Jingchao Zhou, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In CVPR, 2018.

98. Viola and Jones, "Rapid object detection using a boosted cascade of simple features", Computer Vision and Pattern Recognition, 2001

99. D. G. Lowe. Distinctive image features from scale-invariant keypoints. IJCV, 60(2):91–110, 2004

ДОДАТКИ

ДОДАТОК А

```
import os
import time
import urllib.request
from multiprocessing import Queue
from threading import Thread
```

```
from typing import Union

import cv2
import numpy as np
from insightface.app import FaceAnalysis
from insightface.utils import face_align
from lxml import etree
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver import ActionChains, Keys, ChromeOptions
from selenium.webdriver.common.by import By
from tqdm import tqdm

from recognize.views import decode_img


class AppURLopener(urllib.request.FancyURLopener):
    version = 'Mozilla/5.0'


PAUSE = 5
opener = AppURLopener()
htmlparser = etree.HTMLParser()


def goto(element):
    remove_popup()
    actions = ActionChains(driver)
    actions.move_to_element(element).perform()


def load_img_from_src(src):
    if isinstance(src, dict):
        return {k: load_img_from_src(v) for k, v in src.items()}
    if isinstance(src, list):
        return [load_img_from_src(v) for v in src]
    return decode_img(opener.open(src).read())


def get_all_images(tree):
    pictures = tree.xpath()
    urls = [picture.attrib['src'] for picture in pictures]
    name2img = {}

    imgs = []
    for url in urls:
        name = url.split('/')[-1]
        if name not in name2img:
            name2img[name] = load_img_from_src(url)
            imgs.append(name2img[name])



class ImgSaver(Thread):
    def __init__(self):
        super().__init__()
        self.queue = Queue()
```

```
        self.opener = AppURLopener()

        self.app = FaceAnalysis(providers=[
            'CUDAExecutionProvider',
            'CPUExecutionProvider'
        ])
        size = 640
        self.app.prepare(ctx_id=0, det_size=(size, size))

    def save(self, img: Union[str, np.ndarray], folder: str):
        self.queue.put((img, folder))

    def _save(self, img: Union[str, np.ndarray], folder: str):
        if isinstance(img, str):
            img = decode_img(self.opener.open(img).read())
        os.makedirs(folder, exist_ok=True)
        _, kpss = self.app.det_model.detect(img)
        for kps in kpss:
            path2img = os.path.join(folder,
f'{len(os.listdir(folder)):03d}.jpg')
            cv2.imwrite(path2img, face_align.norm_crop(img, landmark=kps))

    def run(self):
        while True:
            img, folder = self.queue.get()
            try:
                self._save(img, folder)
            except ValueError:
                self.save(img, folder)


def remove_popup():
    close_buttons = driver.find_elements(By.XPATH, "//button[@class='pre-
modal-btn-close bg-transparent']")
    if len(close_buttons) and close_buttons[0].is_displayed():
        close_buttons[0].click()


PATH2DATASET = R'D:\faces\men\nike'
PATH2DRIVER = R'D:\Programs\ChromeDriver\chromedriver.exe'

options = ChromeOptions()
options.add_argument('--headless')
driver = webdriver.Chrome(PATH2DRIVER, options=options)
driver.set_window_size(1920, 1080)
driver.maximize_window()
driver.get("https://www.nike.com/w/mens-clothing-6ymx6znik1")
time.sleep(PAUSE)
els = driver.find_elements(By.XPATH, r"//div/p[text()='United States']")
if len(els):
    els[0].click()
    driver.get("https://www.nike.com/w/mens-clothing-6ymx6znik1")
    time.sleep(PAUSE)
img_saver = ImgSaver()
img_saver.start()

products_folder = os.path.join(PATH2DATASET, 'Clothing')
```

```
total_count = int(driver.find_element(By.XPATH, "//span[@class='wall-
header__item_count']").text[1:-1])

tk0 = tqdm(desc='Crawling data', total=total_count)
checked = set()

while True:
    figures = driver.find_elements(By.XPATH, '//figure')

    figure = None
    product_folder = None

    for _figure in figures:
        href = _figure.find_element(By.XPATH, 'a').get_attribute('href')
        idx = href.split('/')[-2]
        product_folder = os.path.join(products_folder, idx)
        if product_folder not in checked:
            checked.add(product_folder)
            tk0.update()
        if not os.path.exists(product_folder):
            goto(_figure)
            figure = _figure
            break

    if figure is None:
        goto(figures[-1])
        continue
    try:
        price = float(
            figure.find_element(By.XPATH, "div/div/div/div/div/div[@data-
test='product-price']").text.replace(' ',

'').replace(
                '$', '')
        )
    except NoSuchElementException:
        os.makedirs(product_folder, exist_ok=True)
        continue

    prod_url_element = figure.find_element(By.XPATH, 'a[@class="product-
card__link-overlay"]')
    prod_url = prod_url_element.get_attribute('href')
    remove_popup()
    ActionChains(driver) \
        .move_to_element(prod_url_element) \
        .key_down(Keys.CONTROL) \
        .click(prod_url_element) \
        .key_up(Keys.CONTROL) \
        .perform()
    time.sleep(PAUSE / 2)
    driver.switch_to.window(driver.window_handles[1])
    time.sleep(PAUSE / 2)
    while True:
        try:
            product_title = \
                [e for e in [e.text for e in driver.find_elements(By.XPATH,
"//h1[@id='pdp_product_title']")] if e][0]
        except IndexError:
```

```
            pass
        else:
            break
    product_headline = \
        [e for e in [e.text for e in driver.find_elements(By.XPATH,
"//h2[contains(@class, 'headline')]")] if e][0]
    version_links_elements = driver.find_elements(By.XPATH,
"//a[contains(@class,'product-overlay')]")
    version_links = [element.get_attribute('href') for element in
version_links_elements]
    if not version_links:
        version_links = [driver.current_url]
    cur_link = np.argmax([version_link == prod_url for version_link in
version_links])
    version_links.insert(0, version_links.pop(cur_link))

    version_idxes = []
    buttons = driver.find_elements(By.XPATH, "//div[@class='mt2-sm css-
12whm6j']/div")

    for url in version_links:
        if url != driver.current_url:
            try:
                s_button = \
                    [e for e, e_url in [(element,
element.get_attribute('href')) for element in version_links_elements]
                     if
                     e_url == url][0]
            except IndexError:
                continue
            goto(s_button)
            s_button.click()
            version_links_elements = driver.find_elements(By.XPATH,
"//a[contains(@class,'product-overlay')]")
            buttons = driver.find_elements(By.XPATH, "//div[@class='mt2-sm
css-12whm6j']/div")

        sizes = {}
        for button in buttons:
            text = button.find_element(By.XPATH, "label").text
            if not text:
                continue
            sizes[text] = button.find_element(By.XPATH,
"input").get_attribute('disabled') is None
        img_elements = []
        while True:
            new_img_elements = driver.find_elements(
                By.XPATH, "//button[contains(@aria-label,'Zoom
in')]//picture/img[not(contains(@alt,'Low Resolution'))]"
            )
            if len(new_img_elements) <= len(img_elements):
                break
            img_elements = new_img_elements
            goto(img_elements[-1])
        version_idx = url.split('/')[-1]
        type_folder = os.path.join(product_folder, version_idx)
        for src in filter(lambda x: x, (element.get_attribute('src') for
element in img_elements)):
```

```
            img_saver.save(src, type_folder)
    os.makedirs(product_folder, exist_ok=True)
    driver.close()
    driver.switch_to.window(driver.window_handles[0])
tk0.close()

print(driver.title)
```

Додаток А – Код для автоматичного збору інформації з сайту Nike

ДОДАТОК Б

```
import numpy as np
from torch.utils.data import Dataset
import os
import cv2


class FeatureInitializer(Dataset):
    def __init__(self, recognition, img_paths, path, dst_path):
        self.recognition = recognition
        self.img_paths = img_paths
        self.path = path
        self.dst_path = dst_path

    def __len__(self):
        return len(self.img_paths)

    def __getitem__(self, item):
        img_path = self.img_paths[item]
        feature_path = f'{os.path.splitext(img_path.replace(self.path,
self.dst_path))[0]}'
        if os.path.exists(f'{feature_path}.npy'):
            features = np.load(f'{feature_path}.npy')
        else:
            img = cv2.imread(img_path)
            if img is not None and tuple(img.shape[:2]) == (112, 112):
                features = self.recognition.get_feat(img)[0]
                os.makedirs(os.path.split(feature_path)[0],
exist_ok=True)
                np.save(feature_path, features)
```

```
                else:
                    features = None
            return features, img_path, feature_path

        @staticmethod
        def collate_fn(batch):
            assert len(batch) == 1
            return batch[0]
```

```
    class FeatureDB:
        def __init__(self, size: int, normalize: bool = True):
            self.size = size
            self.normalize = normalize

            self.values = []
            self.features = None
            self.pointer = 0

        def add(self, features, values):
            if self.features is None:
                self.features = np.empty((self.size, features.shape[0]),
features.dtype)
            elif self.pointer >= self.size:
                self.size *= 2
                new_features = np.empty((self.size, features.shape[0]),
features.dtype)
                new_features[:self.pointer] = self.features
                self.features = new_features
            if self.normalize:
                features = features / np.linalg.norm(features)
            self.features[self.pointer] = features
            self.values.append(values)
            self.pointer += 1

        def get_distances(self, features):
```

```
            return self.features[: self.pointer] @ (features / -
np.linalg.norm(features)) \
                if self.normalize else \
                ((self.features[: self.pointer] - features[None]) **
2).mean(1)

        def __getitem__(self, features):
            return self.values[np.argmin(self.get_distances(features))]

        def get_topk(self, features, topk: int, threshold=0.3):
            distances = self.get_distances(features)
            topk_indices = np.argpartition(distances, min(topk,
distances.shape[0] - 1))[:topk]
            topk_indices = topk_indices[distances[topk_indices] < -
threshold]
            argsort = np.argsort(distances[topk_indices])
            return [[self.values[i], -distances[i]] for i in
topk_indices[argsort]]

        def __repr__(self):
            return str(self)

        def __str__(self):
            try:
                dim = f'{self.features.shape[1]}-dim {self.features.dtype}
'
            except AttributeError:
                dim = ''
            return f'{dim}FeatureDB({self.pointer}/{self.size})'
```

Додаток Б – Класи для роботи з базою даних ознак

ДОДАТОК В

```
import os

import cv2
from insightface.app import FaceAnalysis
from torch.utils.data import DataLoader
from tqdm import tqdm

import numpy as np
```

```
from ai.feature_db import FeatureDB, FeatureInitializer
from utils.files import get_all_files
from insightface.utils import face_align


class Inference:
    def img_path2descr(self, img_path):
        return os.path.splitext(img_path)[0].replace(self.path,
f'{self.path}_descs') + '.txt'

    def __init__(self, topk: int = 100):
        self.topk = topk
        self.app = FaceAnalysis(providers=[
            'CPUExecutionProvider'
        ])
        size = 640
        self.app.prepare(ctx_id=0, det_size=(size, size))
        self.recognition = self.app.models['recognition']

        self.path = r'D:\faces'
        self.dst_path = f'{self.path}_features'

        files = sorted(get_all_files(self.path, ".jpg",
remove_root_folder=False))
        loader = DataLoader(
            FeatureInitializer(self.recognition, files, self.path,
self.dst_path),
            batch_size=1, num_workers=0,
collate_fn=FeatureInitializer.collate_fn
        )

        self.features_db = FeatureDB(len(files))

        for features, img_path, feature_path in tqdm(loader, 'Initializing
DB'):
            if features is not None:
                self.features_db.add(features, img_path)

    def __getitem__(self, img):
        bboxes, kpss = self.app.det_model.detect(img)
        if len(bboxes) > 1:
            argsort = np.argsort(bboxes[:, 0] + bboxes[:, 2])
            kpss = kpss[argsort]
        crops = [face_align.norm_crop(img, landmark=kps) for kps in kpss]
        return [[crop,
*self.features_db.get_topk(self.recognition.get_feat(crop)[0], self.topk)]
for crop in crops]

    def add_to_database(self, img, description=''):
        save_path = os.path.join(self.path, 'added_manually')
```

```
        img_path = os.path.join(save_path,
f'{len(get_all_files(save_path)):06d}.jpg')
        features_path = os.path.splitext(img_path)[0].replace(self.path,
self.dst_path)
        features = self.recognition.get_feat(img)[0]
        if np.min(self.features_db.get_distances(features)) < -.98:
            return
        self.features_db.add(features, img_path)
        os.makedirs(os.path.split(img_path)[0], exist_ok=True)
        cv2.imwrite(img_path, img)
        os.makedirs(os.path.split(features_path)[0], exist_ok=True)
        np.save(features_path, features)
        if description:
            desc_path = self.img_path2descr(img_path)
            os.makedirs(os.path.split(desc_path)[0], exist_ok=True)
            with open(desc_path, 'w+') as f:
                f.write(description)
```

Додаток В – Код для проведення аналізу зображень за допомогою нейромережі

ДОДАТОК Г

```
import datetime
import os
import pickle
import time
import traceback
import base64
import cv2
import jsonpickle
import numpy as np
from flask import Flask, request, Response
from waitress import serve

from ai.module import Inference


def timestamp():
   return
f"[{datetime.datetime.now().replace(microsecond=0).isoformat().replace('T',
' ')}] "


def decode(img):
   if isinstance(img, str):
       img = base64.b64decode(img)
   return cv2.imdecode(np.frombuffer(img, np.uint8), cv2.IMREAD_COLOR)
```

```
if __name__ == '__main__':
    from ai.client import encode
    from recognize.views import opencv2django

    app = Flask(__name__)
    inference = Inference()


    @app.route('/api/add', methods=['POST'])
    def adding_features():
        start = time.perf_counter_ns()
        try:
            r = request
            data: dict = jsonpickle.decode(r.data)
            img = decode(data['img'])
            try:
                description = data['desc']
            except Exception:
                description = ''
            inference.add_to_database(img, description)
            status = 200
            response = {'success': True}
        except Exception as e:
            response = {
                'error': ''.join(traceback.format_exception(None, e,
e.__traceback__)),
                'etype': pickle.dumps(type(e), 0),
                'success': False
            }
            print(response["error"])
            status = 500
        response_pickled = jsonpickle.encode(response)
        print(f'Request processing time: {(time.perf_counter_ns() - start) /
1e6:.2f}ms')
        return Response(response=response_pickled, status=status,
mimetype="application/json")


    @app.route('/api/ai', methods=['POST'])
    def features():
        start = time.perf_counter_ns()
        frames = []
        try:
            r = request
            data: dict = jsonpickle.decode(r.data)
            img = decode(data['img'])
            res = inference[img]
            img_table = []
            text_table = []
            for row in res:
```

```
                _row = []
                text_row = []
                for val in row:
                    if isinstance(val, np.ndarray):
                        img = val
                        color = (0, 0, 0)
                        sim_formatted = f''
                    else:
                        img_path, sim = val
                        img = cv2.imread(img_path) if
os.path.exists(img_path) else None
                        if img is None:
                            continue
                        if sim >= .5:
                            color = (int(255 * (1 - sim) * 2), 255, 0)
                        else:
                            color = (255, max(0, min(int(255 * sim * 2),
255)), 0)

                        sim_formatted = f'{sim:.4f}'
                        desc_path = inference.img_path2descr(img_path)
                        if os.path.exists(desc_path):
                            with open(desc_path, 'r') as f:
                                descritiopn = f.read()
                            sim_formatted =
f'{descritiopn}\n{sim_formatted}'
                    formatted_color = '#' + ''.join(f'0{e}' if len(e) == 1
else e for e in [hex(c)[2:4] for c in color])
                    text_row.append([sim_formatted, formatted_color])
                    _row.append(img)
                img_table.append(_row)
                text_table.append(text_row)

            response = {'imgs': opencv2django(img_table), 'texts':
text_table}
            status = 200
        except Exception as e:
            response = {'error': ''.join(traceback.format_exception(None, e,
e.__traceback__)),
                        'etype': pickle.dumps(type(e), 0)}

            try:
                print(
                    f"{timestamp()}Exception was raised when tried to detect
on frames of shapes: {[getattr(frame, 'shape', 'Failed to get shape!') for
frame in frames]}")

            except Exception:
                print(f"{timestamp()}{response['etype']}Exception was raised
when tried to detect on frames: {frames}")
```

```
            print(response["error"])
            status = 500
        response_pickled = jsonpickle.encode(response)
        print(f'Request processing time: {(time.perf_counter_ns() - start) /
1e6:.2f}ms')
        return Response(response=response_pickled, status=status,
mimetype="application/json")


    DEBUG = False
    if DEBUG:
        app.run(host="localhost", port=8080)
    else:
        serve(app, host="localhost", port=8080)
```

Додаток Г – Код для створення програмного інтерфейсу для взаємодії з базою даних і нейромережою з іншого процесу/комп’ютеру

ДОДАТОК Ґ

```
import time

import cv2
import jsonpickle
import numpy as np
import requests


def encode(img):
    if isinstance(img, np.ndarray) and img.ndim == 3:
        return cv2.imencode('.jpg', img)[1].tobytes()
    return img


class AIClient:
    def __init__(self, ip, port=8080):
        self.ip = ip
        self.port = port
        self.url = f'http://{ip}:{port}/api'

    def send(self, img, max_wait: float = 3.):
        start = time.monotonic()
        while True:
            try:
                response = requests.post(
                    f'{self.url}/ai', data=jsonpickle.encode({
                        'img': encode(img)
                    }))
            except requests.exceptions.RequestException as e:
                if not max_wait or time.monotonic() - start < max_wait:
```

```
                    time.sleep(0.1)
                else:
                    raise TimeoutError('Could not get response from server
in time') from e
            else:
                break
        return jsonpickle.decode(response.content)

    def add_image(self, img, description='', max_wait: float = 3.):
        start = time.monotonic()
        while True:
            try:
                response = requests.post(
                    f'{self.url}/add', data=jsonpickle.encode({
                        'img': encode(img), 'desc': description
                    }))
            except requests.exceptions.RequestException as e:
                if not max_wait or time.monotonic() - start < max_wait:
                    time.sleep(0.1)
                else:
                    raise TimeoutError('Could not get response from server
in time') from e
            else:
                break
        return jsonpickle.decode(response.content)
```

Додаток Ґ – Код для використання програмного інтерфейсу для взаємодії з базою даних