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)
Герга Ілля Антонович
(прізвище, ім'я, по батькові)

Керівник _Краснощок Віктор Миколайович
(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)

Київ – 2021

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	26.10.2020	
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	заява
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	05.03.2021	
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	09.04.2021	
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	07.05.2021	
9.	Подання роботи у першому варіанті	11.05.2021	
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**24.05.2021**	
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	
14.	Захист кваліфікаційної роботи бакалавра	23.06.2021	

Здобувач вищої освіти _____

(підпис)

Керівник _____

(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Зміст пояснювальної записки (перелік питань під час дослідження)

Складові частини дипломної роботи	Обсяг, арк. 62
Титульний аркуш	1 ст.
Завдання до дипломної роботи (календарний план проекту)	1 ст.
Відомість дипломної роботи	1 ст.
Пояснювальна записка до дипломної роботи	1 ст.
Анотація	1 ст.
Анотація (іноземною мовою-англійською)	1 ст.
Зміст	2 ст.
Словник термінів	1 ст.
Вступ	3 ст.
1. Загальносистемні питання. Аналіз розв’язуваної задачі й огляд наявних результатів. Постановказадачі та проектування	17 ст.
2. Проектні і технічні рішення. Види забезпечення	15 ст.
3. Опис роботи програми	10 ст.
Висновки	3 ст.
Перелік посилань	2 ст.
Додатки	6 ст.

				ДП ХХХХ 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.				Відомість дипломної роботи	Лист	Листів
Керівн.						
Н/контр.	Макаренко С.А.					
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ

Дипломна робота: 62 с., 26 рис., 1 табл., 15 джерел, 4 дод.

Дана дипломна робота присвячена створенню динамічних веб-застосунків із використанням бібліотеки OpenCV.

Для розробки оптимізованої системи було проведено дослідження всіх актуальних, на даний час, технологій, що дозволяють виявляти, аналізувати та розпізнавати обличчя. Після досліджень було здійснено порівняння найпопулярніших бібліотек та сервісів, їх швидкість розпізнавання та простоту використання для звичайного користувача.

Вибір OpenCV для дипломної роботи був зроблений по трьом основним критеріям: швидкість завантаження, програмне супроводження системи та вирішення проблем які виникнуть при розробці веб-застосунку.

Для розробки веб-застосунку розпізнавання облич було використано одну з найбільш популярних та найкращих бібліотек - OpenCV.

Метою роботи є ефективне та швидке розпізнавання облич людей.

Об'єктом дослідження у цій роботі є процес розпізнавання обличчя.

Предметом дипломної роботи є програмні засоби розпізнавання обличчя.

Методи дослідження. Було використано при дослідженні такі наукові методи: аналіз, синтез, аналогія, узагальнення, моделювання, абстрагування, індукція, системний аналіз і синтез при побудові програмної системи.

Ключові слова: веб-застосунок, Python, OpenCV, HTML.

ANNOTATION

This thesis is devoted to the creation of dynamic web applications using the OpenCV library.

To develop an optimized system, a study of all current technologies that allow to detect, analyze and recognize faces was conducted. The research compared the most popular libraries and services, their speed of recognition and ease of use for the average user.

The choice of OpenCV for the thesis was made on three main criteria: download speed, software support of the system and solving problems that will arise during the development of a web application.

One of the most popular and best libraries, OpenCV, was used to develop a face recognition web application.

The purpose of my thesis is: effective and fast recognition of people's faces.

The object of study in this work is the process of face recognition.

The subject of study is facial recognition software.

Research methods. The following scientific methods were used in the study: analysis, synthesis, analogy, generalization, modeling, abstraction, induction, systems analysis and synthesis in building a software system.

Keywords: web application, Python, OpenCV, HTML.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ШІ – штучний інтелект

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту.

OpenCV (англ. Open Source Computer Vision Library) - бібліотека комп'ютерного зору з відкритим кодом

CSS (Cascading Style Sheets) — спеціальна мова, яку використовують для графічного опису веб-сайтів

SPA (англ. single-page application) - це веб-застосунок чи веб-сайт, який вміщується на одній сторінці

DeepFace - це система з глибоким навчанням система розпізнавання осіб створена дослідницькою групою в Facebook

FaceNet - нейронна мережа, яка вчиться перетворювати зображення особи в компактний евклідовий простір

CPU (англ. central processing unit) - електронний блок або інтегральна схема, яка виконує машинні інструкції

API (англ. Application Programming Interface) – це набір чітко визначених методів для взаємодії різних компонентів

ORM (англ. Object-relational mapping, Об'єктно-реляційна проекція) — технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних»

БД – база даних

JSON (JavaScript Object Notation) – текстовий формат обміну даними, заснований на JavaScript

URL (Uniform Resource Locator) – єдиний вказівник на ресурс

PCA (англ. Principal component analysis) - один з основних способів зменшити розмірність даних, втративши найменшу кількість інформації

AR (англ. augmented reality) — термін, що позначає всі проєкти, спрямовані на доповнення реальності будь-якими віртуальними елементами

Anaconda - відкрито розповсюджуваний дистрибутив різних програмних продуктів, зокрема, мов програмування Python та R

SVC (англ. Support Vector Machine) - метод аналізу даних для класифікації та регресійного аналізу за допомогою моделей з керованим навчанням

ROC (англ. Receiver Operating Characteristic) - графік, що дозволяє оцінити якість бінарної класифікації

AUC (англ. Area Under The ROC Curve) - міра якості моделі бінарної класифікації

ВСТУП

Актуальність дослідження. Повсюдне розпізнавання облич полегшує складні завдання. Автоматичний вхід по телефону, простіший доступ до біометричних даних, швидка ідентифікація біля захисних воріт, покупки (наприклад, купівля курки) і навіть пошук коханого вашої мрії. Сьогодні ми можемо використовувати розпізнавання обличчя у всіх можливих ситуаціях. Вперше ідея технології розпізнавання обличчя з'явилася в 2014 році, коли Facebook оголосив про запуск DeepFace. Лише для запису програма може визначити з точністю 97,25%, чи належать два сфотографовані обличчя одній людині. Потім, через рік, у 2015 році, Google покращився завдяки FaceNet, досягнувши нового рекорду - 99,63%.

Зараз, у 2021 році, більшість алгоритмів розпізнавання облич є кращими за найточніші алгоритми наприкінці 2013 року. Ось чому ми можемо легко використовувати його без сильної команди інженерів, нам потрібні лише відповідні інструменти.

Дана робота допомагає досліджувати та аналізувати проблеми, пов'язані з розробкою веб-додатків, а також допомагає вибрати найбільш зручну та найшвидшу технологію для створення веб-сайтів.

Практична частина дипломної роботи представлятиме собою проектування та розроблення веб-застосунку, який зможе розпізнати людину використовуючи машинне навчання.

Метою роботи є ефективне та швидке розпізнавання облич людей.

Завдання дослідження:

- висвітлити теоретичні основи концептуальних підходів щодо розпізнавання облич;
- здійснити аналіз наявних інструментів створення веб-застосунків та надати щодо них переваги і недоліки;

- дослідити стек технологій щодо створення веб-застосунку розпізнавання облич;
- спроектувати, розробити та впровадити веб-застосунок розпізнавання облич.

Об’єктом дослідження у цій роботі є процес розпізнавання обличчя.

Предметом дипломної роботи є програмні засоби розпізнавання обличчя.

Методи дослідження. Було використано при дослідженні такі наукові методи: аналіз, синтез, аналогія, узагальнення, моделювання, абстрагування, індукція, системний аналіз і синтез при побудові програмної системи.

Практичне значення одержаних результатів. Система створена з урахуванням та виключенням недоліків які були присутніми в інших програмах-прототипах. Дозволяє в автоматизованому режимі дізнатися всі необхідні дані.

Значимість дослідження полягає в наступному: розглянуто можливості штучного інтелекту як технології розпізнавання облич.

Структура роботи. Дипломна робота складається зі вступу, трьох розділів, поділених на підрозділи, висновків, додатків та списку використаних джерел з 14 найменувань.

РОЗДІЛ 1: АНАЛІЗ ВИКОРИСТАННЯ ВЕБ-ЗАСТОСУНКУ ПРИ РОЗПІЗНАВАННІ ОБЛИЧЧЯ

1.1 Технологія комп’ютерного зору

Велику частину інформації людина отримує завдяки зору. Звідси розвиток технологій комп'ютерного зору і розпізнавання об'єктів, є найбільш значущою для розвитку штучного інтелекту (ІІ), яке також є одним з основних призначень програмування.

Напрямок розпізнавання осіб ж є приватною гілкою в технології розпізнавання об'єктів. Тому, розвиток даного напрямку незмінно призведе до розвитку всієї технології розпізнавання об'єктів.

Незважаючи на велику різноманітність алгоритмів розпізнавання осіб, можна виділити загальну структуру даного процесу, яка представлена на рис. 1.1.

Рисунок 1.1 Загальна структура розпізнавання осіб

На першому етапі проводиться виявлення і локалізація особи на зображенні (найбільш ефективним є використання методу Віоли-Джонса). Стеження має на увазі більш спрощені способи локалізації особи (т. К. Обличчя до даного моменту однозначно визначено) на наступних кадрах безперервного відео.

На другому етапі проводиться вирівнювання зображення в знайденій області (геометричне і яскраве перетворення, застосування фільтрів). Обчислення і порівняння ознак варіюється між методами, і при цьому все зводиться до певного порівнянні обчислених ознак з закладеними в базу даних еталонами. У даній роботі буде також розглянуто використання методу гнучкого порівняння на графах.

1.2 Веб-застосунок та його складові

Веб-програми - це прикладне програмне забезпечення, в якому клієнт є браузером, а сервер - веб-сервером. На відміну від веб-сайтів, веб-програми є динамічними та виконують певні завдання програми. Сьогодні веб-програми дуже поширені і ними користуються мільйони людей щодня. Вони поєднують досвід створення зручних графічних інтерфейсів для настільних та мобільних додатків з досвідом легкого доступу до них із будь-якого веб-браузера. Одні лише ці переваги забезпечують величезний попит на веб-програми для широкого кола підприємств. Flask - це мікрофреймворк для веб-додатків, створений за допомогою Python. Він базується на наборі інструментів Werkzeug та механізмі шаблонів Jinja2. Поширюється на умовах ліцензії BSD. Flask використовується для розробки таких проектів, як сторінки Pinterest, LinkedIn та Flask.

Flask називається мікрофреймворком, оскільки для нього не потрібні спеціальні інструменти або бібліотеки. Йому бракує рівня абстракції для роботи з базами даних, перевірки форми чи інших компонентів, які забезпечують широко використовувані функції через сторонні бібліотеки. Однак Flask підтримує розширення, які надають додаткові атрибути, як якщо б вони були доступні в Flask з самого початку. Існують розширення декількох поширених інструментів, що використовуються для встановлення зв'язків між об'єктами, перевірки форм, контролю процесу завантаження та підтримки

різних технологій та платформ відкритої автентифікації. Розширення оновлюються частіше, ніж вихідний код. Для того, щоб найкраще визначити зовнішній вигляд веб-сайту, розробники покладаються на можливості веб-браузера, який певною мірою дозволяє контролювати зовнішній вигляд, а не його вміст. Завдяки мові розмітки гіпертексту HTML розробники отримали таку можливість.

HTML (мова розмітки гіпертексту) - це стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сайтів створюються за допомогою HTML (або XHTML). Веб-браузер обробляє документ HTML і відтворює його на екрані користувача звичайним способом. HTML дозволяє визначати заголовки, абзаци, таблиці, зображення та інші матеріали, а потім веб-браузер інтерпретує ці теги. Наприклад, в одному веб-браузері теги веб-сторінок будуть розпізнані з початку абзацу та представлятимуть веб-сторінку у бажаній формі, тоді як в іншому браузері вся веб-сторінка буде знаходитися в одному рядку.

Всі теги HTML можна розділити на дві категорії:

1. Визначте, як веб-браузер відображатиме теги всього тіла документа;

2. Теги, що описують основні атрибути веб-сторінки, такі як автор або заголовок веб-сторінки.

Веб-сторінки можливо бути створити за допомогою будь-якого текстового редактора або редакторів коду та конвертерів. Вибір редактора, який буде використовуватися для написання HTML-документів, залежить виключно від зручності роботи з ним.

З іншого боку, більшість звичайних засобів для написання коду мають влаштовані конвертери, це дозволяє конвертувати звичайний документ в HTML веб-сторінку.

Основна перевага HTML являється в тому, що будь-яку веб-сторінку можливо переглянути в веб-браузерах різного типу та на різних платформах.

CSS (Cascading Style Sheets) — спеціальна мова, яку використовують для графічного опису веб-сайтів, написаних на мові HTML. Найбільш частіше CSS використовують для візуальної презентації веб-сторінок, написаних HTML(або XHTML), але формат CSS можливо застосовувати також і до інших видів XML-документів.

CSS можливо додати прямо в HTML-сторінку – це називається внутрішніми таблицями стилів. Також CSS можна описати в окремому файлі, та приєднати посилання на файл до потрібної HTML-сторінки - це називається зовнішніми таблицями стилів. Зовнішню таблицю потрібно підключити до основного HTML-документу за допомогою спеціальних тегів: <link rel="stylesheet" href="style.css >, де style.css - це ім'я css файлу, що містить таблицю CSS. В такому випадку використовувати описаний у зовнішній таблиці CSS можливо використовувати повторно безліч разів.

З попередньої інформації можна зробити наступні висновки, тобто використання HTML та CSS зручно та просто у використанні, але цей метод має багато недоліків, таких як:

- Багато коду;
- Неможливо створити динамічні сторінки;
- Обмежена функціональність веб-сайту.

1.3 Переваги та недоліки веб-застосунку в порівнянні з десктопними програмами

Ставлення сучасних користувачів до веб-додатків досі неоднозначне. Чим вони кращі за настільні комп’ютери, і чим вони в нестатку? Спробуємо це зрозуміти.

Переваги браузерних програм очевидні.

По-перше, користувачам не потрібно встановлювати важке програмне забезпечення на свої машини. Для повноцінної роботи потрібен лише браузер (зазвичай входить в операційну систему) та доступ до Інтернету.

По-друге, встановлюючи програми на свій комп'ютер, мимоволі береш на себе обов'язки адміністратора, що доставляє багато клопоту для недосвідчених користувачів. Вам потрібно встановити та запустити програму, потім налаштувати її самостійно, а потім раптово з'являться випадкові помилки, які потрібно негайно вирішити. Для браузерних програм, які насправді розташовані на сервері, вам не доведеться про це турбуватися. По суті, роль адміністратора веб-додатків - це розробник, який працює в одному місці. Наприклад, у корпоративному відділі набагато вигідніше та ефективніше підтримувати команду програмістів та адміністраторів для встановлення та налаштування настільних програм на машинах користувачів.

По-третє, веб-програми вимагають ресурсів і не вимагають жодних вимог до апаратної платформи. Це означає, що не має значення, скільки мегабайт оперативної пам'яті встановлено на комп'ютері користувача та з якої операційної системи він працює. Для браузерів та доступу до Інтернету все інше не так важливо.

Крім того, немає проблем із підтримкою та зворотною сумісністю старої версії програми. Коли з'являється нова версія настільного додатка, користувачам зазвичай доводиться вирішувати проблеми, пов'язані з оновленням копії, вже встановленої на їх комп'ютері. У випадку браузерної програми така проблема не виникає - існує лише одна версія, і всі користувачі працюють, а у випадку з новою версією вони автоматично перемикаються на неї без винятку, іноді навіть не помічаючи цього. І нарешті, веб-застосунки дозволяють своїм користувачам бути по-справжньому мобільними. По суті, ви можете працювати в мережі, зберігати результати своєї роботи на сервері і, в разі необхідності, мати до них доступ звідусіль, де є вихід в Інтернет.

На жаль, у веб-додатків також є слабкі сторони. Звичайно, ці слабкі сторони не затьмарюють їх сильних сторін. Але останнє вже не виглядає таким привабливим у їхньому контексті.

Перш за все, на жаль, Інтернет зараз доступний, але не скрізь - принаймні в нашій країні. І все ще є багато недоліків у багатьох аспектах наших "величезних" витрат на трафік та широти Інтернет-каналів. Крім того, існує велика кількість програм, які неможливо замінити програмами браузера (принаймні найближчим часом). Наприклад, ви не можете створювати складні 3D-моделі в браузері.

Нарешті, головний недолік веб-додатків - багатьох користувачів бентежить той факт, що їх дані будуть зберігатися та оброблятися десь на чужому сервері. Зрештою, це може призвести до витоку, втрати або спотворення інформації (в деяких випадках це може). Не кожен ризикне розмістити особисту інформацію в Інтернеті.

1.4 Бібліотека OpenCV

OpenCV - загальна бібліотека функцій та алгоритмів для комп'ютерного зору, обробки зображень та числових алгоритмів з відкритим кодом. Бібліотека надає інструменти для обробки та аналізу вмісту зображень, включаючи ідентифікацію об’єктів на фотографіях (таких як обличчя та символи, текст тощо), відстеження руху об’єктів, перетворення зображень, застосування методів машинного навчання та виявлення загальних елементів на фотографіях. Різні зображення.

Бібліотека розроблена компанією Intel і в даний час підтримується Willow Garage та Itseez. Вихідний код бібліотеки написаний на C ++ і поширюється за ліцензією BSD. Прив'язки підготовлені для різних мов програмування, таких як Python, Java, Ruby, Matlab, Lua тощо. Його можна безкоштовно використовувати в академічних та комерційних цілях. Проект

OpenCV був офіційно запущений в 1999 році за ініціативою Інституту досліджень Intel для розробки додатків, що потребують великої кількості процесорів.

Основними вкладниками проекту була Intel's Performance Library Team та певна кількість експертів з чисельної оптимізації у Inter Russia. На перших етапах розвитку OpenCV основними задачами бібліотеки були:

- Розробити дослідження в напрямку комп'ютерного зору та забезпечити добре оптимізовані бібліотеки з відкритим кодом.
- Поширення знань у галузі комп'ютерного зору, забезпечення загальної інфраструктури, яку можуть розробляти розробники, та спрощення сприйняття та обміну кодом.
- Розробити комерційні комерційні програми та створити незалежну від платформи, оптимізовану та безкоштовну бібліотеку. Для цього використовується ліцензія, яка не вимагає відкриття таких комерційних програм.

Перша альфа-версія OpenCV була представлена на конференції IEEE Computer Vision and Image Recognition у 2000 році, а п'ять бета-версій були випущені між 2001 і 2005 роками. Перша версія 1.0 була випущена в 2006 році. У середині 2008 року OpenCV отримав корпоративну підтримку від Willow Garage і відновив активний розвиток. "Попередня версія" версії 1.1 вийшла в жовтні 2008 року.

Друга основна версія OpenCV була випущена в жовтні 2009 року. OpenCV 2 включає основні зміни в інтерфейсі C ++. Ці зміни спрямовані на більш прості, безпечні для моделі, додавання нових функцій та кращу реалізацію існуючих моделей з точки зору продуктивності (особливо в багатоядерних системах). Офіційний випуск продовжує виходити кожні 6 місяців, розроблений незалежною командою з Росії та підтримуваний комерційними компаніями.

У серпні 2012 року підтримка OpenCV була передана некомерційній організації OpenCV.org.

Бібліотека містить понад 2500 алгоритмів оптимізації, включаючи повний набір класичних та практичних алгоритмів машинного навчання та комп'ютерного зору. Алгоритм OpenCV використовується в наступних областях:

- Аналіз та обробка зображень
- Системи розпізнавання обличчя
- Ідентифікації об'єктів
- Розпізнавання жестів на відео
- Відстежування переміщення камери
- Побудова 3D моделей об'єктів
- Створення 3D хмар точок зі стерео камер
- Склеювання зображень між собою, для створення зображень всієї сцени з високою роздільною здатністю
- Система взаємодії людини з комп’ютером
- Пошуку схожих зображень із бази даних
- Усування ефекту червоних очей при фотозйомці зі спалахом
- Стеження за рухом очей
- Аналіз руху
- Ідентифікація об'єктів
- Сегментація зображення
- Трекінґ відео
- Розпізнавання елементів сцени і додавання маркерів для створення доповненої реальності та інші.

1.5 Обрана мова програмування

Розвиток мови програмування Python розпочався наприкінці 1980-х. Ця мова була задумана наприкінці 1980-х років, а реалізована в грудні 1989 року голландським дослідником Гідо ван Россумом. Python - нащадок мови програмування ABC, має можливість обробляти винятки та помилки, а також здатність взаємодіяти з операційною системою Amoeba. Напрямок розвитку Python насправді можна простежити з його назви, воно було винайдено спільнотою розробників. Ця мова сьогодні широко використовується в багатьох галузях і є мовою програмування високого рівня. Його дизайн та філософія підкреслюють читабельність коду. У порівнянні із використанням C ++ або Java, синтаксис дозволяє коротше описувати завдання. Мова може реалізовувати структури для побудови чітких і зрозумілих програм незалежно від масштабу.

Python реалізує різноманітні парадигми програмування, включаючи об’єктно-орієнтовану, імперативну та функціональну. Він має динамічну систему типу та збирач сміття. Однією з найбільших переваг Python є те, що майже у всіх сферах існує велика кількість відкритих і доступних бібліотек, які постійно оновлюються та розвиваються за рахунок величезної спільноти. Інтерпретатор Python можна встановити на багатьох операційних системах, що дозволяє використовувати програми, написані на ньому, у різних системах. У Python ви можете використовувати інші парадигми, такі як проектування контрактів та логічне програмування, а також зовнішні розширення. Python використовує динамічний введення тексту та круговий пошук сміття для управління пам’яттю. Ще однією особливістю Python є пізнє прив'язування, яке пов'язує імена методів та змінних під час виконання. Дизайн Python дозволяє розробляти програми у функціональному стилі, дотримуючись традицій Lisp.

Мова має вбудовані функції фільтрації, роботи з кортежами, списками, генераторами випадквоих значень. Стандартна бібліотека має два модулі itertools та functools. Також Python має значні переваги перед різними мовами програмування, наприклад:

- Чистий синтаксис, дозволяє розбивати програму на окремі блоки та модулі
- Довільний стиль написання прогами (що характерно для більшості інтерпретованих мов)
- Принцип роздільного створення модулів передбачає змогу викорасання тільки необхідних елеметів і мінімальну кількість написаного коду
- використання Python в інтерактивному режимі (дуже корисно для експериментів та вирішення простих проблем)
- наявність великої кількості бібліотек для візуалізації гафічного інтерфейсу і даних
- підходить для розв'язання математичних задач (реалізація операцій з комплексними числами)
- з цілими числами довільної величини, командний рядокможе використовуватися як потужний калькулятор).

Однак у Python все ж є деякі недоліки. Python, як і в багатьох інших інтерпретованих мовах прогамування, де не застосовуються, JIT-компілятори мають загальний недолік - відносно низьку швидкість виконування програми. Крім того, відсутність статичного набору тексту та деякі інші фактори, на жаль, не дозволяють, під час компіляції, реалізувати механізм перевантаження функцій.

Окрім добре продуманої та збалансованої мови програмування, Python також широко використовується в різних сферах. Це може бути для створення сценаріїв та запуску незалежних програм для різних компонентів. Як універсальна мова, python розвивається у багатьох сферах - від розробки веб-

сайтів та ігор до управління роботами та космічними кораблями. Програми Python можуть шукати файли та дерева каталогів, запускати інші програми та паралельно обробляти процеси та потоки. Стандартна бібліотека Python оснащена прив'язками POSIX, розширеннями файлів, утилітами zip-файлів, аналізаторами XML та JSON, обробниками файлів CSV тощо. Крім того, більшість системних інтерфейсів Python придатні для інтеграції; наприклад, сценарії, які зазвичай копіюють дерева каталогів, залишаються однаковими на всіх основних платформах Python.

Python постачається зі стандартним об'єктно-орієнтованим інтерфейс API Tk GUI під назвою tkinter (Tkinter в 2.X), що дозволяє програмам Python реалізовувати портативні графічні інтерфейси з авторською реалізацією. Графічні інтерфейси Python / tkinter працюють без змін у Microsoft Windows, Linux та Mac OS. Крім того, пропонує графічний інтерфейс API wxPython, розроблений на основі бібліотек C ++

Набори інструментів вищого рівня, такі як Dabo, побудовані на основі базових API, таких як wxPython та tkinter. За допомогою відповідної бібліотеки ви також можете використовувати підтримку GUI в інших наборах інструментів у Python, такі як Qt з PyQt, GTK з PyGTK, MFC з PyWin32, .NET з IronPython та Swing з Jython або JPype.

1.6 Приклад тренування моделі машинного навчання

Ми будемо використовувати OpenCV, бібліотеку з відкритим кодом для комп'ютерного зору, написану на C / C ++, яка має інтерфейси на C ++, Python та Java. Він підтримує Windows, Linux, MacOS, iOS та Android. Деякі наші роботи також потребуватимуть використання Dlib, сучасного набору інструментів C ++, що містить алгоритми машинного навчання та інструменти для створення складного програмного забезпечення.

```
import cv2
import matplotlib.pyplot as plt
import dlib
from imutils import face_utils


font = cv2.FONT_HERSHEY_SIMPLEX
```

Рисунок 1.2 Підключення пакетів

Спочатку потрібно зрозуміти, що таке каскадні класифікатори. Каскадний класифікатор, а саме каскад підсилених класифікаторів, що працюють з haar-подібними ознаками, є особливим випадком навчання ансамблю, який називається підсиленням. Зазвичай він покладається на класифікатори Adaboost (та інші моделі, такі як Real Adaboost, Gentle Adaboost або Logitboost).

Каскадні класифікатори навчаються на декількох сотнях зразків зображень, що містять об'єкт, який ми хочемо виявити, та інших зображень, що не містять цих зображень.

Як ми можемо виявити, чи є там обличчя чи ні? Існує алгоритм, який називається фреймворк виявлення об’єктів Віола – Джонс, який включає всі кроки, необхідні для виявлення обличчя в реальному часі:

- Haar Feature Selection, функції, похідні від вейвлетів Хаара
- Створення цілісного образу
- Навчання Adaboost
- Каскадні класифікатори

Є деякі загальні риси, які ми знаходимо на найбільш поширених людських обличчях:

- темна область очей порівняно з верхніми щоками;
- яскрава область перенісся в порівнянні з очима;
- якесь конкретне розташування очей, рота, носа.

Дані характеристики називаються рисами обличчя Haar. Процес вилучення об’єкта буде виглядати так:

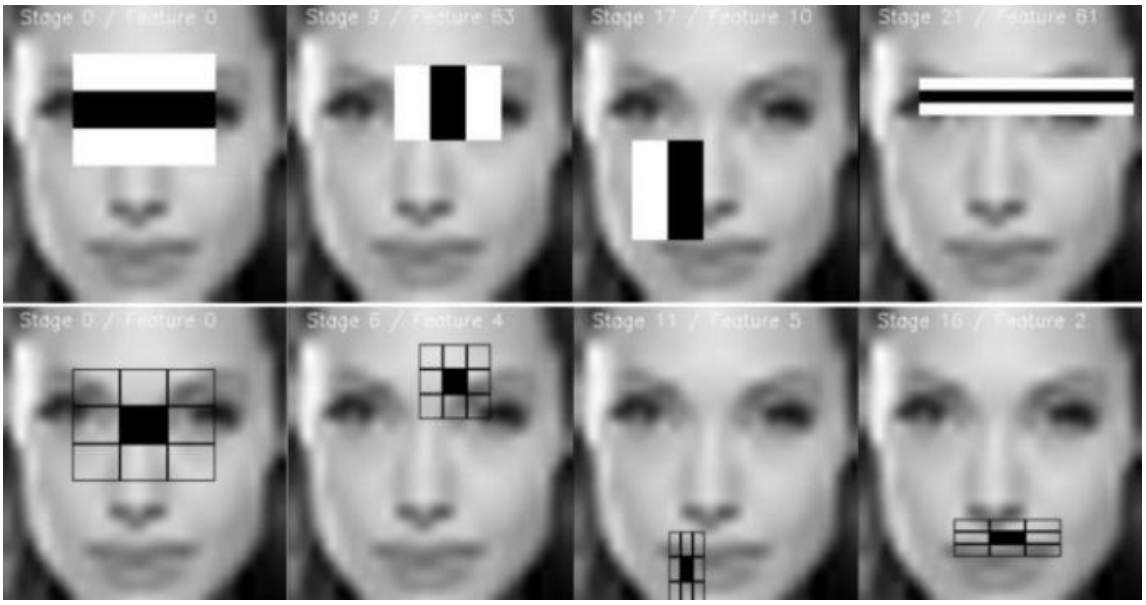

Рисунок 1.3 Риси обличчя Haar

У цьому прикладі перша ознака вимірює різницю в інтенсивності між областю очей та областю на верхніх щоках. Значення функції просто обчислюється шляхом підсумовування пікселів у чорній області та віднімання пікселів у білій області.

Ми можемо завантажити тестове зображення:

```
# Load the image
gray = cv2.imread('face_detect_test.jpeg', 0)


plt.figure(figsize=(12,8))
plt.imshow(gray, cmap='gray')
plt.show()
```

Рисунок 1.4 Код завантаження зображення

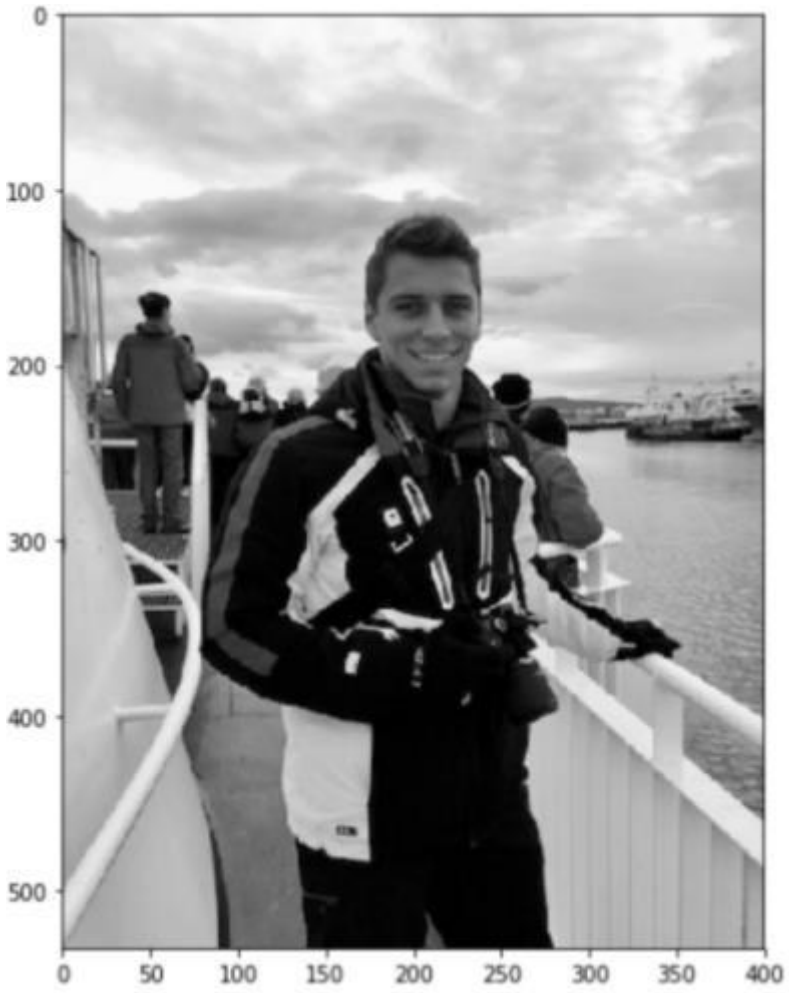

Рисунок 1.5 Тестове зображення

Потім ми виявляємо обличчя і додаємо навколо нього прямокутник:

```
# Detect faces
faces = faceCascade.detectMultiScale(
gray,
scaleFactor=1.1,
minNeighbors=5,
flags=cv2.CASCADE_SCALE_IMAGE
)


# For each face
for (x, y, w, h) in faces:
    # Draw rectangle around the face
    cv2.rectangle(gray, (x, y), (x+w, y+h), (255, 255, 255), 3)
```

Рисунок 1.6 Виявлення обличчя

Ось список найпоширеніших параметрів функції detectMultiScale:

- ScaleFactor: Параметр, що визначає, наскільки зменшений розмір зображення при кожному масштабі зображення.
- minNeighbors: параметр, який вказує, скільки сусідів повинен мати кожен прямокутник-кандидат, щоб зберегти його.
- minSize: мінімально можливий розмір об'єкта. Об’єкти менших розмірів ігноруються.
- maxSize: максимально можливий розмір об'єкта. Об'єкти, більші за них, ігноруються.

Нарешті, відображаємо результат:

```
plt.figure(figsize=(12,8))
plt.imshow(gray, cmap='gray')
plt.show()
```

Рисунок 1.7 Відобразити знайдене обличчя

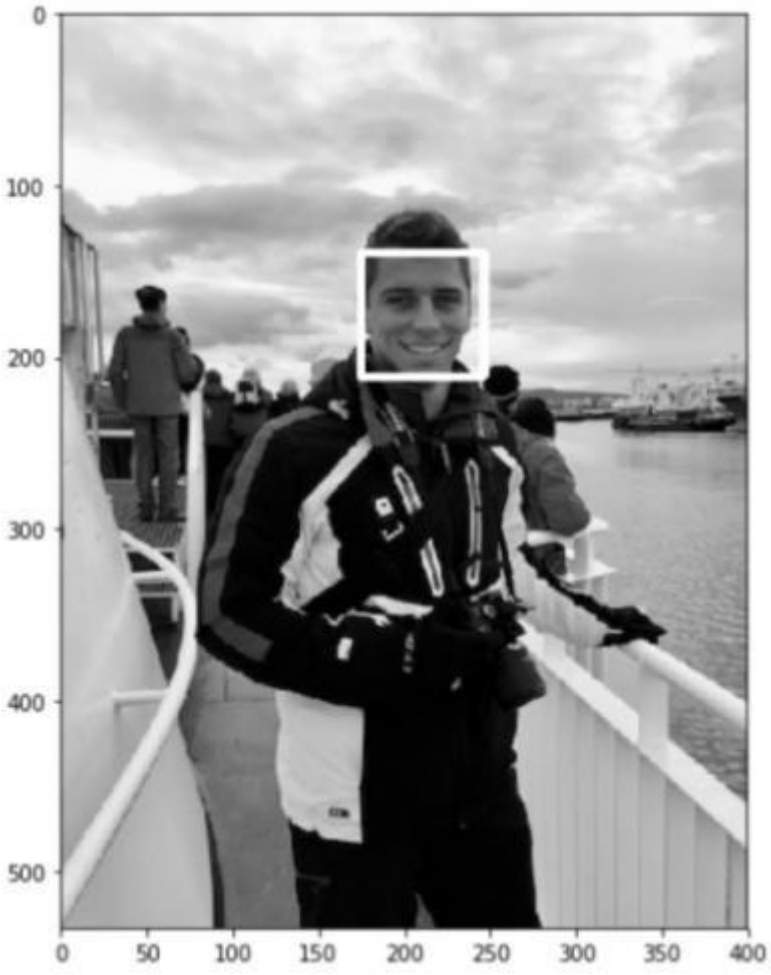

Рисунок 1.8 Результат пошуку обличчя

1.7 Власні інтерфейси з PCA

Один із методів зменшення розмірності називається аналізом основних компонентів (PCA). Ідея PCA полягає в тому, що ми хочемо вибрати гіперплощину таким чином, щоб, коли всі точки проектуються на неї, вони були максимально розподілені. Іншими словами, нам потрібна вісь максимальної дисперсії. Давайте розглянемо наш приклад сюжету нижче. Потенційною віссю є вісь x або вісь y, але в обох випадках це не найкраща вісь. Однак, якщо ми вибрамо лінію, яка прорізає наші дані по діагоналі, це вісь, де дані будуть найбільш поширені.

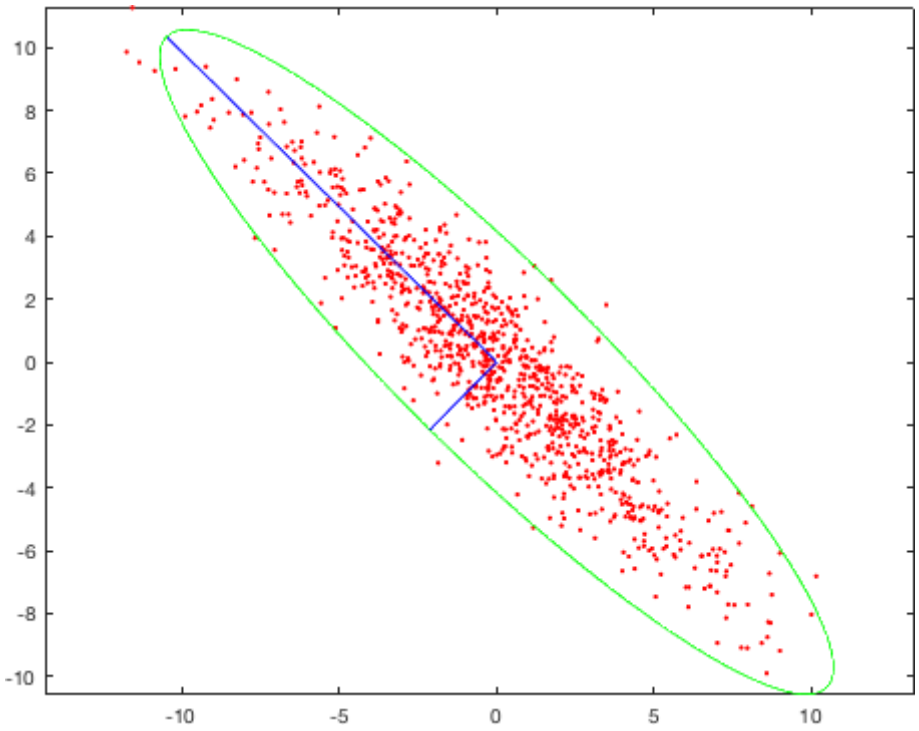

Рисунок 1.9 Аналіз основних компонентів

Більша синя вісь - правильна вісь. Якщо ми спроектуємо наші точки на цю вісь, вони будуть розподілені якомога більше. Але як ми розуміємо цю вісь? Ми можемо запозичити термін з лінійної алгебри, який називається власними векторами! Це їх власний інтерфейс і вони отримали свою назву. По

суті, ми обчислюємо матрицю коваріації даних і розглядаємо найбільший власний вектор матриці коваріації. Це наша основна вісь, на яку ми проектуємо дані для зменшення розмірності. За допомогою цього методу ми можемо отримати високовимірні дані та зменшити їх до нижчих розмірів, вибравши найбільші власні вектори матриці коваріації та спроектувавши на них власні вектори.

Оскільки ми обчислюємо осі максимального поширення, ми зберігаємо найважливіші аспекти наших даних. Нашому класифікатору легше відокремлювати обличчя, коли наші дані розподіляються, а не згруповані.

Як це пов’язано з нашим викликом розпізнавання обличчя? Ми можемо концептуалізувати наші m * n зображення як точки в m*n-мірному просторі. Тоді ми можемо використовувати PCA, щоб зменшити наш простір від m*n до чогось набагато менше. Це допоможе пришвидшити наші обчислення і буде стійким до шуму та змін.

Висновки

Для кращого розуміння існуючих методів побудови сучасних веб-додатків, я проаналізував реалізацію веб-додатків у порівнянні з десктопними програмами та їх переваги та недоліки. Що стосується технологій, він розглянув найпопулярніші фреймворки та бібліотеки, які з'явилися на ринку веб-розробки та використовуються у сучасних продуктах.

На підставі отриманих знань, інформації про проблематику та огляду технологій було прийнято рішення про формулювання постановки задачі для подальшої роботи над застосунком.

Підсумовуючи зазначу, що з огляду на технології та сучасні тренди у розробці фінальний варіант постановки задачі має вигляд веб-застосунку, що базується на принципі Single page application.

РОЗДІЛ 2: ВИБІР ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ВЕБ-ЗАСТОСУНКУ РОЗПІЗНАВАННЯ ОБЛИЧЧЯ

Відповідно до постановки завдання необхідно провести аналіз та обґрунтувати вибір технологій та інструментів для розробки програмного забезпечення з урахуванням специфіки веб-проектування та інтеграції з хмарним середовищем. Оскільки в сучасній веб-розробці сайти давно перестали нести статично відображувану інформацію в проекті буде передбачено клієнтську частину (фронтенд), серверну частину (бекенд) та реалізовано базу даних із подальшим розміщенням на хмарному ресурсі.

2.1 Розробка застосунку з використанням фреймворку Flask

Flask - це ліцензована мікрорамка BSD, заснована на Werkzeug та Jinja2. Характеристика мікрокадрів полягає в тому, що вони намагаються надати розробникам лише ті компоненти, які необхідні для досягнення поставленого завдання. Мікрокадри можуть бути спеціально розроблені для створення API для певних служб або сайтів. Flask досить простий, але в той же час дуже гнучкий і дозволяє розробникам використовувати лише необхідну конфігурацію, що полегшує розробку програм або плагінів. Колба була розроблена компанією Pocoo і відразу ж доступна. Сьогодні фреймворк підтримується проектом The Pallets, який розробляє нові компоненти та модернізує старі. Варто зазначити, що Flask має дуже активну спільноту розробників та кілька великих форумів. Розгортання та моніторинг API є важливою частиною REST API. У Flask сама парадигма розробки також зміниться відповідно до розширення API.

Двома основними компонентами Flask є Werkzeug та Jinja2. Хоча Werkzeug відповідає за забезпечення маршрутизації, налагодження та інтерфейсу шлюзу веб-сервера (WSGI), механізм шаблонів - Jinja2. Сама Flask не підтримує доступ до бази даних, автентифікацію користувача чи будь-які

інші розширені утиліти, але підтримує велику кількість розширень для досягнення вищезазначених функцій. Простий додаток можна навіть реалізувати в одному файлі, але при реалізації великого додатка найкраще розподілити програму по модулях. Модульна структура також є однією з переваг Flask. Основною ідеєю цього фреймворку є досягнення міцної основи для програм, а функції верхнього рівня відображають розширення.

Flask-спільнота досить велика та активно займається розробкою сотень розширень, які одразу публікуються у відкритий доступ. Команда Flask core постійно маніторить нові розширення та гарантує, що затверджені розширення сумісні з майбутніми випусками. Будучи мікрофреймворком, Flask забезпечує розробникам гнучкість у виборі дизайнерських рішень, котрі чудово вписуються в архітекру проекту. Також розробники ведуть реєстр розширень, який регулярно оновлюється та постійно підтримується.

Flask, як і всі інші бібліотеки Python, можна встановити, використовуючи індекси пакетів Python (PPI), і його дуже просто налаштувати і почати розробляти.

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, From Flask!'

if __name__ == '__main__':
    app.run()
```

Рисунок 2.1 Приклад використання фреймворку

Даний код імпортує бібліотеку Flask, ініціює додаток, створивши екземпляр класу Flask, оголошує маршрут, а потім визначає функцію для виконання при виклику маршруту. Цього коду достатньо для запуску першої програми Flask. Цей код запускає дуже простий вбудований сервер, котрий чудово пыдходить для тестування, але недостатньо потужний для введеня додатку в експлуатацію.

Як уже обговорювалося, Flask не здійснює підтримку доступу до бази даних, і для здійснення взаємодії з БД, здебільшого використовують розширення Flask під назвою Flask-SQLAlchemy, що надає підтримку бібліотеки SQLAlchemy. По суті, SQLAlchemy - це набір інструментів Python SQL та Object Relational Mapper, що забезпечує розробникам повну потужність і гнучкість SQL.

SQLAlchemy повністю підтримує шаблони проектування на рівні підприємства, прагнучи досягти ефективного доступу до баз даних, зберігаючи при цьому ефективність та простоту використання. Хорошим висновком у розробці додатків є реалізація модулів автентифікації користувачів, CRUD (створення, читання, оновлення та видалення даних) та API REST для створення, пошуку, маніпулювання та видалення об'єктів. Flask також дозволяє інтегрувати утиліти Swagger для створення документації API, написання тестів та їх інтеграції. Для вузького функціонального тестування прийнято використовувати pytest, який є повнофункціональним інструментом для тестування програм Python. Pytest спрощує розробку тестів, а бібліотека досить розширювана для підтримки складних випадків використання. Postman - це зріла платформа REST API, яка забезпечує інтегровані інструменти для кожного етапу життєвого циклу API, що робить розробку API простішою та надійнішою.

Реалізація фронтенду на проекті потребує обов’язкового використання класичних інструментів веб-розробки, таких як мова гіпертекстової розмітки HTML5 та каскадної таблиці стилів CSS3.

За сучасними тенденцями переважна більшість веб-застосунків являють собою SPA, тобто односторінковий застосунок, контент якого динамічно змінюється завдяки використанню скриптової мови програмування JavaScript. Порівнюючи зі звичними «багатосторінковими» сайтами, SPA надає переваги у швидкості, адже не потребує перезавантаження всієї сторінки, а змінює лише необхідні блоки сайту без зайвих затримок. На сьогоднішній день існує

широкий вибір технологій, що спрощують розробку односторінкових веб-застосунків.

Також був використаний **Bootstrap**. Це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків.

Bootstrap — це клієнтський фреймворк, тобто інтерфейс для користувача, на відміну від коду серверної сторони, який знаходиться на сервері. Репозиторій із цим фреймворком є одним із найпопулярніших на GitHub.

2.2 Аналіз інструментів та засобів розробки бекенд частини веб-застосунку

Python - інтерпретована мова програмування загального призначення високого рівня. Філософія дизайну Python підкреслює читабельність коду через очевидне використання великих просторів. Його мовна структура та об’єктно-орієнтований підхід покликані допомогти програмістам писати чіткий та логічний код для малих та великих проектів.

Django (Джанго) - відкритий фреймворк Python високого рівня (програмний фреймворк) для розробки веб-систем. Сайт на Django побудований з однієї або декількох частин, і рекомендується модулювати його. Це одна із суттєвих архітектурних відмінностей між цим фреймворком та іншими фреймворками, такими як Ruby on Rails. Архітектура Django схожа на модель-вигляд-контроллер (MVC). Однак так званий "контролер" у класичній моделі MVC у Django називається "видом", а те, що має бути "видом", називається "шаблоном". Тому розробники MVC називають Django MTV ("модель-шаблон-вигляд").

Деякі можливості Django:

- ORM, API доступу до БД з підтримкою транзакцій;
- вбудований інтерфейс адміністратора, з уже наявними перекладами на більшість мов;
- диспетчер URL на основі регулярних виразів;
- розширювана система шаблонів з тегами та наслідуванням;
- система кешування;
- інтернаціоналізація;
- архітектура застосунків, що підключаються, які можна встановлювати на будь-які Django-сайти;
- «generic views» - шаблони функцій контролерів;
- авторизація та аутентифікація, підключення зовнішніх модулів аутентифікації: LDAP, OpenID та ін.;
- система фільтрів («middleware») для побудови додаткових обробників запитів, наприклад включені в дистрибутив фільтри для кешування, стиснення, нормалізації URL і підтримки анонімних сесій;
- бібліотека для роботи з формами (наслідування, побудова форм за існуючою моделлю БД);
- вбудована автоматична документація по тегам шаблонів та моделям даних, доступна через адміністративний застосунок;

Різні компоненти фреймворку між собою пов'язані слабо, тому достатньо будь-яку частину замінити на аналогічну. Наприклад, замість вбудованих шаблонів можна використовувати Mako або Jinja

Jinja — рушій шаблонів для мови програмування Python створений Арміном Ронакером з ліцензією BSD. На відміну від схожого рушія шаблонів у Django, використовує вирази у стилі мови Python та використовує пісочницю для шаблонів. Завдяки тому, що шаблони Jinja засновані на текстовому форматі, тому створення розмітки документу стає подібним до написання сирцевого коду.

Шаблони рушія Jinja надають можливості налаштування тегів, фільтрів, тестів та глобальних параметрів. Також, на відміну від рушія Django, Jinja дозволяє розробнику шаблонів викликати функції з об'єктами у якості аргументів. Jinja є основним рушієм шаблонів у Flask.

NumPy - це основний пакет, необхідний для наукових обчислень з Python. Він забезпечує:

- потужний N-вимірний об'єкт масиву
- складні (мовні) функції
- інструменти для інтеграції коду C / C ++ та Fortran
- корисна лінійна алгебра, перетворення Фур'є та можливості випадкових чисел

pandas - це пакет для мови програмування Python, який забезпечує швидкі, гнучкі та виразні структури даних, робить операції з “реляційними” або “позначеними” даними одночасно простими та інтуїтивно зрозумілими. Він має на меті стати фундаментальним будівельним елементом високого рівня для практичного аналізу даних у реальному світі на Python. Крім того, він має більш широку мету - стати найпотужнішим та найгнучкішим інструментом аналізу / маніпулювання даними з відкритим кодом, доступним будь-якою мовою. Вона вже на шляху до цієї мети.

Ось декілька речей, які pandas добре виконує:

- Легка обробка відсутніх даних (представлених як NaN, NA або NaT) як з плаваючою точкою, так і з даними без плаваючої точки
- Змінюваність розміру: стовпці можна вставляти та видаляти з DataFrame та об’єктів більшого розміру
- Автоматичне та явне вирівнювання даних: об’єкти можуть бути явно вирівняні за набором міток, або користувач може просто ігнорувати мітки і дозволити Series, DataFrame та іншим структурам автоматично вирівнювати дані для вас під час обчислень

- Потужна, гнучка група за функціональністю для виконання операцій розділення-застосування-об'єднання над наборами даних, як для агрегування, так і для перетворення даних
- Спрощення перетворення нерівних, по-різному індексованих даних в інших структурах даних Python та NumPy в об’єкти DataFrame
- Інтелектуальне нарізування на основі етикеток, вигадливе індексування та підмножина великих наборів даних
- Інтуїтивне об’єднання наборів даних
- Гнучка зміна форми та обертання наборів даних
- Ієрархічне маркування осей (можна мати кілька міток на галочку)
- Надійні інструменти введення / виводу для завантаження даних із плоских файлів (CSV та з роздільниками), файлів Excel, баз даних та збереження / завантаження даних із надшвидкого формату HDF5
- Функціональні можливості часових рядів: генерація діапазону дат та перетворення частоти, статистика переміщення вікон, зміщення та відставання дат

2.3 Опис структури системи

Структура системи - колекція елементів системи та взаємозв'язок між ними у формі колекції. Структура системи відноситься до структури, розташування, порядку та відображає певні взаємозв'язки. Розташування компонентів системи, тобто її обладнання, не враховує багато атрибутів (станів) її елементів.

На схемі на рис. 2.2 продемонстровано структуру веб-застосунку на етапі розробки

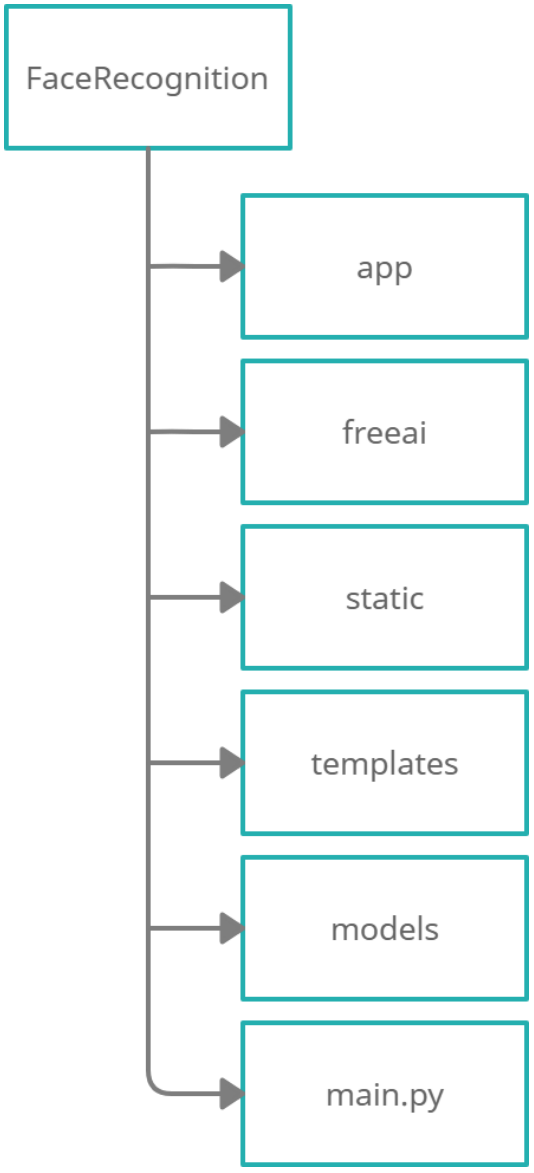

Рисунок 2.2 Структура директорій програми розпізнавання

Саме ці директорії виконують головну роль у створенні веб-застосунку. Кожна з них відповідає за окрему частину проекту. Детальний опис директорій можна побачити в таб. 2.1.

Таблиця 2.1

Опис директорій програми FreeAI

Назва директорії/файла	Опис
__pycache__	Директорія, яка містить Python файли, скомпільовані в байт-код
app	Директорія, яка містить модулі для роботи з файлами
models	Директорія, яка містить треновані моделі та модулі для роботи з ними
templates	Директорія, яка містить файли html сторінок застосунку
static	Директорія, яка містить додаткові файли стилів та зображень, які відображаються на сайті
main.py	Файл, який використовується для запуску сервера застосунку
init_models.py	Скрипт, який використовується для тренування моделей
freeai	Директорія з різними бібліотеками для машинного навчання

2.4 Інструмент для розробки моделі машинного навчання – Jupyter notebook

Jupyter Notebook - неймовірно потужний інструмент для інтерактивної розробки та подання проектів в області наук про дані. У цій статті ви дізнаєтеся, як налаштувати Jupyter Notebooks на локальному комп'ютері і як почати використовувати його в ваших проектах.

Почнемо з визначення: що таке «notebook» (блокнот)? Блокнот об'єднує код і його висновок в єдиний документ, який об'єднує візуалізацію, розповідний текст, математичні рівняння та інші мультимедійні. Цей інтуїтивно зрозумілий робочий процес сприяє итеративной і швидкій розробці, що робить ноутбуки все більш популярним вибором для подання в даних і їх аналізу.

Найкраще те, що в рамках проекту з відкритим вихідним кодом Project Jupyter він повністю безкоштовний.

Проект Jupyter є наступником більш раннього проекту IPython Notebook, який вперше був опублікований в якості прототипу в 2010 році. Хоча в Jupyter Notebooks можна використовувати з багатьма різними мовами програмування, в цій статті основну увагу буде приділено Python, оскільки він є найбільш поширений варіантом використання.

Щоб отримати максимальну віддачу від цього уроку, ви повинні бути знайомі з програмуванням, особливо з Python і pandas. Проте, якщо у вас є досвід роботи з іншою мовою, Python в цій статті не буде занадто складним, а стаття все одно буде вам корисною в налаштуванні Jupyter Notebooks локально. Як ви побачите пізніше в цій статті, Jupyter Notebooks також може виступати в якості гнучкою платформи для роботи з pandas і навіть з Python.

У Windows ви можете запустити Jupyter за допомогою ярлика, який Anaconda додає в ваше меню «Пуск», яке відкриє нову вкладку в браузері за замовчуванням, і яка повинна виглядати приблизно так, як показано на наступному рисунку:

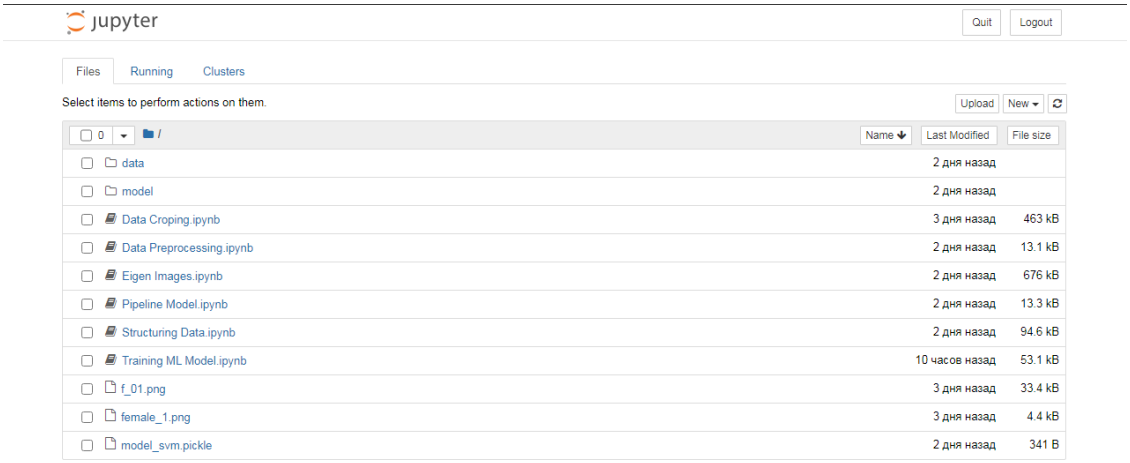

Рисунок 2.3 Панель інструментів програми

2.5 Метод опорних векторів

У машинному навчанні метод опорного вектора - це метод аналізу даних, який використовує керовані моделі навчання та пов'язані з ними алгоритми навчання для класифікаційного та регресійного аналізу. Він називається опорною векторною машиною (SVM), також звана опорною векторною машинною мережею, англійська підтримка векторної мережі. Для даного набору навчальних зразків кожен зразок позначається як приналежний до тієї чи іншої з двох категорій.Алгоритм навчання OVM будує модель і присвоює новий зразок тій чи іншій категорії, роблячи його порядком Неймовірний двійковий лінійний класифікатор. Модель IWM представляє зразки як точки в просторі, відображаючи, що зразки з кожної категорії розділені найширшим чистим простором. Потім зіставте нові зразки в той самий простір і передбачте, до якої категорії вони належать, виходячи з того, в яку сторону розриву вони потрапляють.

На додаток до лінійної класифікації, IBA також може ефективно виконувати нелінійну класифікацію, використовуючи так звані звукові прийоми, неявно відображаючи їх вхід у високомірний простір ознак. Коли дані не марковані, управління навчанням неможливе, і потрібно спонтанне

навчання, намагаючись природним чином згрупувати дані в групи, а потім зіставити нові дані з цими сформованими групами. Удосконалений алгоритм машинного векторного кластеризації, який називається кластером опорних векторів, зазвичай використовується в промислових додатках, коли дані не марковані або лише деякі дані позначені як раніше оброблені перед проходженням класифікації.

ОВМ може бути використаний для вирішення багатьох практичних завдань:

- ОВМ може бути використаний для класифікації тексту та гіпертексту, оскільки їх використання може значно зменшити потребу у маркуванні навчальних зразків у стандартних налаштуваннях індукції та трансдукції.
- Ви також можете використовувати комп’ютер для класифікації зображень. Експериментальні результати показують, що після трьох-чотирьох раундів зворотного зв’язку IBA може досягти вищої точності пошуку, ніж традиційні рішення для уточнення запитів. Те саме стосується систем сегментації зображень, у тому числі тих, які використовують модифіковані версії комп'ютерів, що використовують привілейований метод, запропонований Вапником.
- За допомогою комп’ютера можна розпізнавати рукописні символи.
- Алгоритм OVM широко використовується в біології та інших наукових галузях. Вони використовуються для класифікації білків, правильно класифікуючи до 90% інгредієнтів.

2.6 Дистрибутив Anaconda Python

Anaconda має розширений модуль підключення (понад 1500), включаючи власний менеджер Conda та віртуального середовища. Графічний

інтерфейс, Anaconda Navigator як графічна альтернатива інтерфейсу командного рядка (CLI).

Найбільша різниця між менеджерами пакетів Conda та Pip полягає в тому, як керувати залежностями пакету з'єднань, що є основною проблемою при використанні Data Science у Python та основною причиною появи Conda.

Коли Pip встановлює пакунки, які бажає клієнт, він автоматично встановить весь список залежних пакетів Python, не перевіряючи, чи конфліктують вони з раніше встановленими пакетами. Через це правильно встановлений користувач, такий як Google Tensorflow, може виявити, що він раптово перестає працювати: Pip встановить іншу версію NumPy при встановленні нового пакету (для цього потрібні встановлений пакет та існуючий Tensorflow), наприклад 3.6, і Tensorflow може працювати нормально лише з 3.5. У деяких випадках може здатися, що новий пакет працює на перший погляд належним чином, але насправді це дасть результат, відмінний від правильного результату, який видно лише в деяких деталях.

На противагу цьому, Conda структурно аналізує все поточне програмне середовище, а потім встановлює новий пакет, беручи до уваги всі обмеження сумісності, різні версії пакунків, а точніше, щоб забезпечити набір комбінованих пакетів. У деяких випадках Conda попереджатиме користувачів, що певні пакети не можна використовувати одночасно. Таким чином, тепер користувачі можуть мати, наприклад, Tensorflow 2.0 або новішу версію, і вибрати зручний варіант, розуміючи особливості кожної версії пакета.

Ви можете використовувати команду conda install, щоб встановити пакети з відкритим кодом окремо від сховища Anaconda, Anaconda Cloud або власного сховища чи дзеркала.

У Навігаторі за замовчуванням доступні такі програми:

- JupyterLab
- Jupyter Notebook
- QtConsole

- Spyder
- Glueviz
- Orange
- Rstudio
- Visual Studio Code

2.7 Проведення тестування веб-ресурсу

Фінальний результат веб-застосунку призначений для розпізнавання обличчя. Після реалізації даного веб-застосунку було проведено тестування роботи сайту. Сайт однаково добре відображається у браузерах, таких як Google Chrome, Mozilla Firefox, Opera та інші.

Створений веб-дизайн виглядає просто та сучасно.

Контактна інформація в нижній частині веб-застосунку має зручний функціонал для редагування.

Всі сторінки та блоки з текстовою інформацією, які розташовані, мають єдиний стиль, шрифт та розмір.

Сайт має сучасний вигляд, працює швидко. Тобто отримані результати повністю відповідають поставленим задачам.

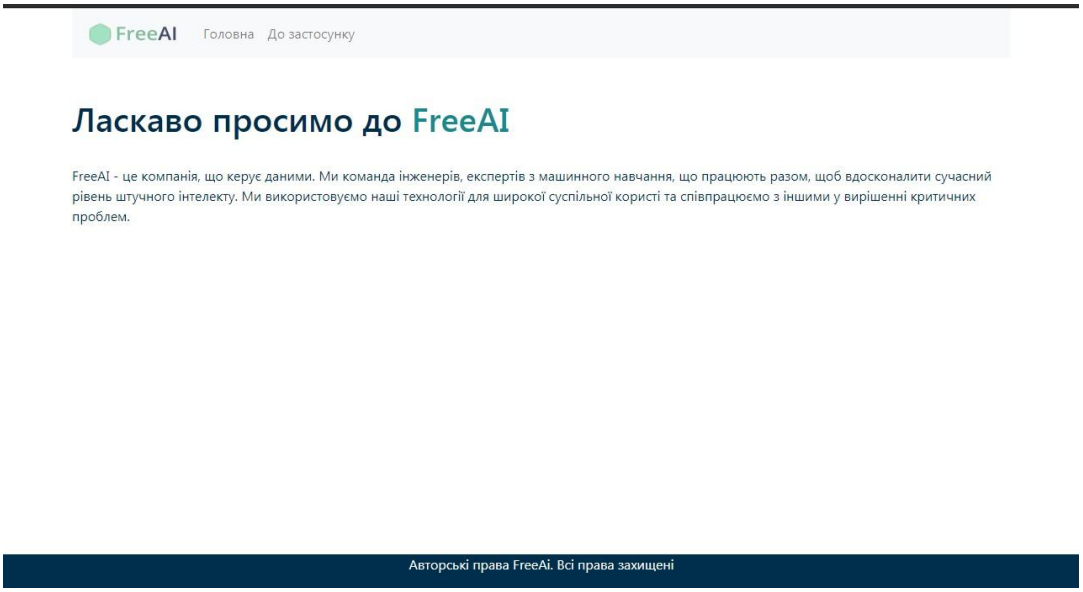

Рисунок 2.4 Головна сторінка веб-застосунку

Висновки

У цьому розділі, відповідно до поставленого технічного завдання, було проведено обґрунтування стеку технологій для побудови веб-застосунку розпізнавання обличчя.

На основі інформації, отриманої в попередньому розділі, я вирішив використовувати мову розмітки гіпертексту HTML як основну структуру та мову програмування Python.

Також, для ефективнішої роботи із стилізації було використано Bootstrap, бібліотеку для розробки інтерфейсу користувача який є чудовим доповненням до стилю застосунку.

РОЗДІЛ 3: ОПИС РОБОТИ СИСТЕМИ

3.1 Специфікація веб-застосунку

Веб-застосунок розпізнавання обличчя створений для аналізу облич та подальшого розпізнавання. Зараз, у 2021 році, більшість алгоритмів розпізнавання облич перевершують найточніший алгоритм кінця 2013 року. Ось чому ви можете легко використовувати його без надпотужної команди інженерів, а все, що вам потрібно - це лише відповідні інструменти.

Розпізнавання обличчя має три стадії:

Виявлення - процес пошуку обличчя на зображенні. Це можна зробити, навчивши алгоритм, як правило, глибоку нейронну мережу, на величезній кількості фотографій, які мають обличчя у відомих місцях. Завдяки Flickr, Instagram або Facebook ми маємо готові масивні набори зображень, які використовуються для навчання глибоких нейронних мереж.

Аналіз (атрибуція) - це етап, який часто відображає обличчя шляхом вимірювання відстані між вузловими точками, що включає взаємозв'язок між очима, носом, бровами та іншими рисами обличчя. Потім ці вимірювання транскрибуються в однокодовану модель, яку потім можна порівняти та потенційно поєднати з відомими фотографіями в базі даних.

Розпізнавання - це спроба підтвердити особу людини на фото. Це останній крок, який має на меті дати остаточну відповідь на питання - Хто на цій картині?

Робота мого застосунку полягає у розпізнаванні людини за характеристиками її обличчя.

3.2 Розробка програмного застосунку

При побудові веб-застосунку розпізнавання обличчя не обійтись без моделі машинного навчання. Існує багато способів її побудови та варіацій налаштування. Зокрема, розпізнавання очей, тіла, посмішки та інше. Для своєї роботи я побудував та натренував модель розпізнавання обличчя. Всі етапи побудови можна побачити на рисунку 3.1.

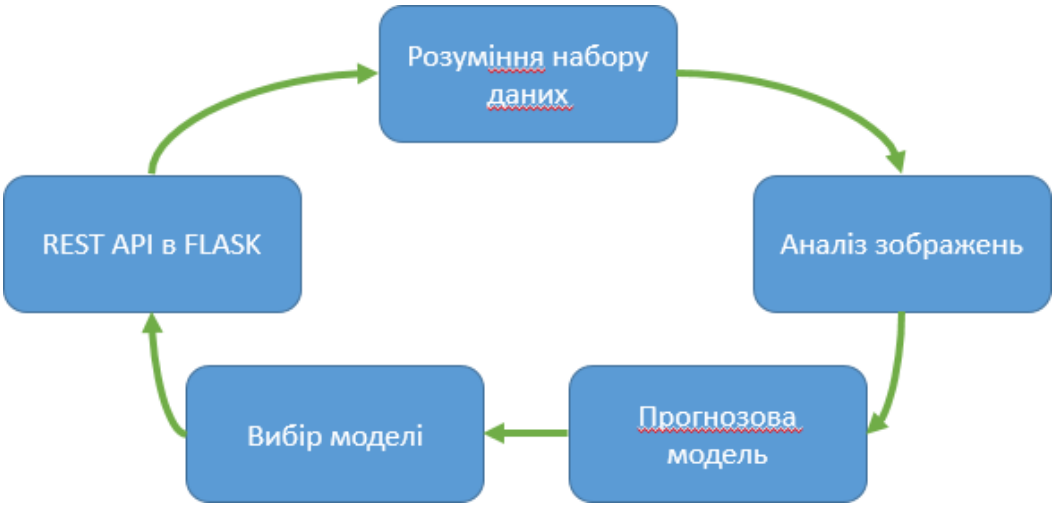

Рисунок 3.1 Етапи побудови моделі машинного навчання

Для зручності, побудуємо план з коротким описом кожного етапу:

1) Розуміння набору даних
   a) Обробка зображень за допомогою opencv
2) Аналіз зображень
   a) Попередня обробка даних
   b) Аналіз дослідницьких даних
3) Прогнозова модель
   a) Eigen-зображення за допомогою методу опорних векторів
   b) Тренування моделі машинного навчання
4) Вибір моделі

a) Оцінка моделі

b) Тюнінг моделі

5) REST API в Flask

a) Інтерфейс шлюзу веб-сервера

b) Інтегрування прогнозуючої моделі

Далі, нам потрібно знайти набір даних з обличчями на базі якої ми будемо тренувати модель машинного навчання. Для своєї роботи я використав безкоштовний dataset взятий з сайту data.vision.

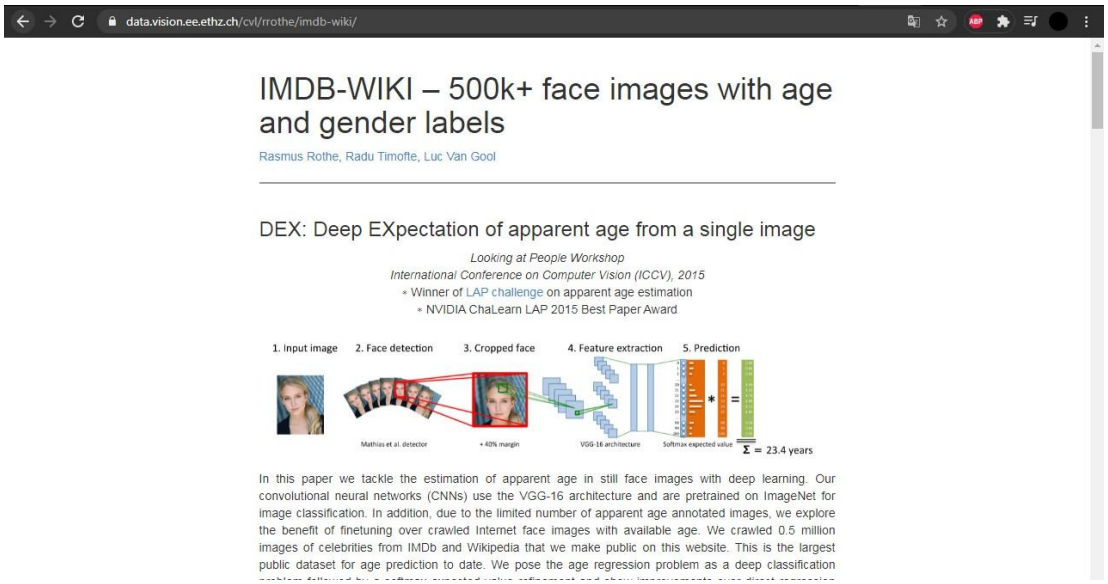

Рисунок 3.2 Сервіс з набором даних

Отримані дані я помістив в окрему папку.

Рисунок 3.3 Репозиторії отриманих даних

Головною ідеєю було те, що нам потрібно було взяти зображення з папки data_images, обрізати обличчя та помістити отримані файли в папку

crop_images. Ми виділяли область в якій знаходиться саме обличчя та скористалися моделлю Haar Cascade Classifier.

```
Ввод [18]: # Load haar cascade classifier
           haar = cv2.CascadeClassifier('./model/haarcascade_frontalface_default.xml')

Ввод [37]: faces = haar.detectMultiScale(gray,1.1,2)
           print(faces)

           [[ 70  56 126 126]]

Ввод [29]:

           ()

Ввод [38]: for x,y,w,h in faces:
               cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
           plt.imshow(img)

Out[38]: <matplotlib.image.AxesImage at 0x78eeeb0>
```

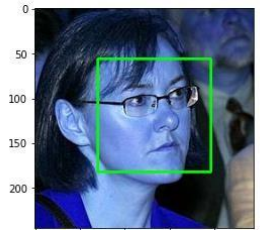

Рисунок 3.4 Приклад використання моделі Haar Cascade Classifier

Далі потрібно було обрізати отриману область та виконати даний алгоритм для всіх зображень.

```
Ввод [39]: # crop the image
           crop_img = img[y:y+h,x:x+h]

Ввод [40]: plt.imshow(crop_img)

Out[40]: <matplotlib.image.AxesImage at 0x7830970>
```

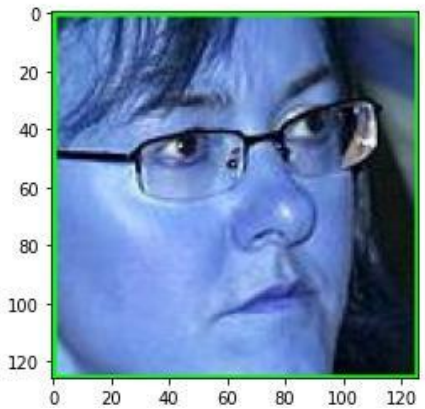

Рисунок 3.5 Приклад обрізаного зображення

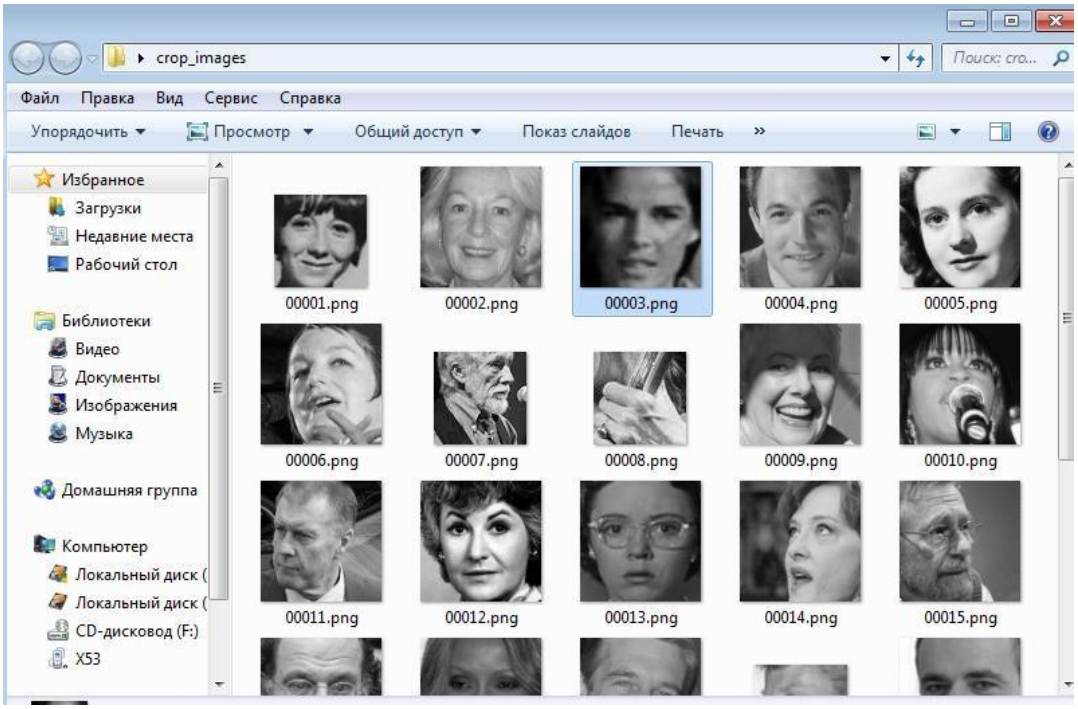

Рисунок 3.6 Вміст папки crop_images

На даному етапі ми маємо неструктуровану базу зображень, яку в майбутньому ми перетворимо. За допомогою цих даних буде можливою побудова моделі машинного навчання.

```
# hyper parameter tuning
```

```
model_tune = SVC()
```

```
from sklearn.model_selection import GridSearchCV
```

```
param_grid = {'C':[1,10,20,30,50,100],
              'kernel':['rbf','poly'],
              'gamma':[0.1,0.05,0.01,0.001,0.002,0.005],
              'coef0':[0,1],
              }
```

```
model_grid = GridSearchCV(model_tune,param_grid=param_grid,scoring='accuracy',cv=5,verbose=1)
```

```
model_grid.fit(X,y)
```

```
Fitting 5 folds for each of 144 candidates, totalling 720 fits

GridSearchCV(cv=5, estimator=SVC(),
             param_grid={'C': [1, 10, 20, 30, 50, 100], 'coef0': [0, 1],
                         'gamma': [0.1, 0.05, 0.01, 0.001, 0.002, 0.005],
                         'kernel': ['rbf', 'poly']},
             scoring='accuracy', verbose=1)
```

```
model_grid.best_params_
```

```
{'C': 30, 'coef0': 0, 'gamma': 0.001, 'kernel': 'rbf'}
```

```
model_grid.best_score_
```

```
0.7853211009174312
```

Рисунок 3.7 Кінцеве налаштування моделі

3.3 Інструкції користувача

Отже, нас зустрічає головна сторінка з коротким описом веб-застосунку та проблем які він вирішує (рис 3.2). Щоб перейти до застосунку, натиснемо кнопку «До застосунку» у верхній панелі управління.

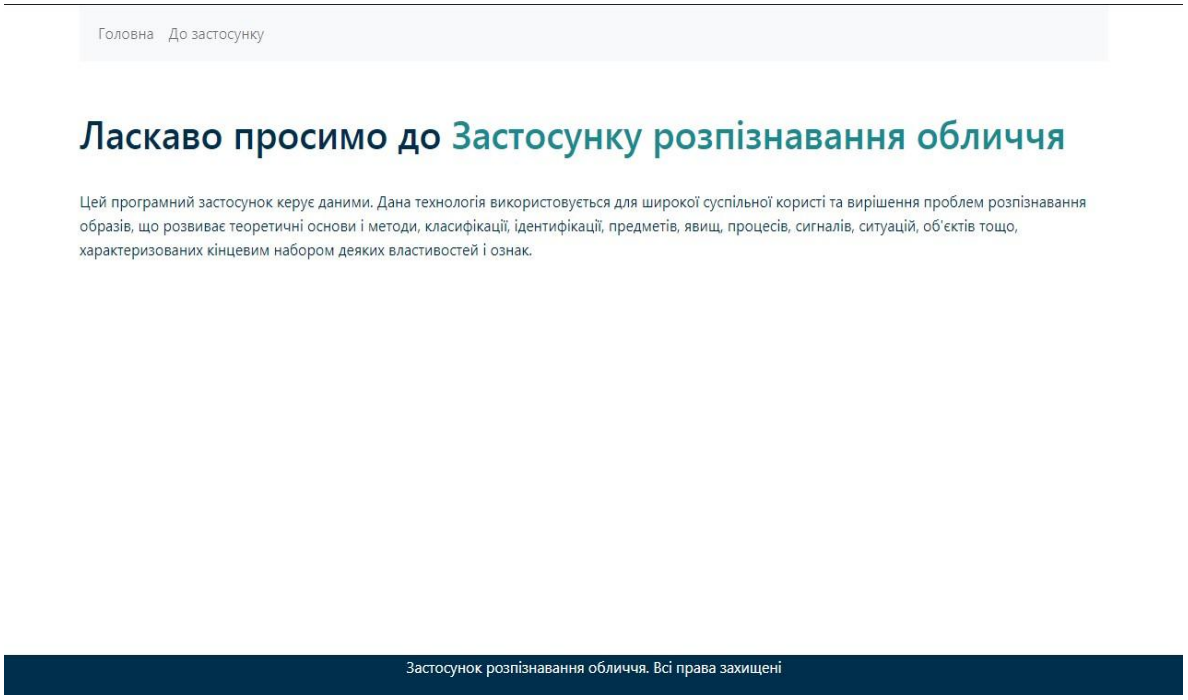

Рисунок 3.8 Головна сторінка

Відкрилась сторінка з описом роботи веб-застосунку та загальним малюнком. Після ознайомлення з наданою інформацією, ми можемо перейти до вкладки з самою програмою розпізнавання, як показано на рис 3.3, та натиснути кнопку «Спробувати»:

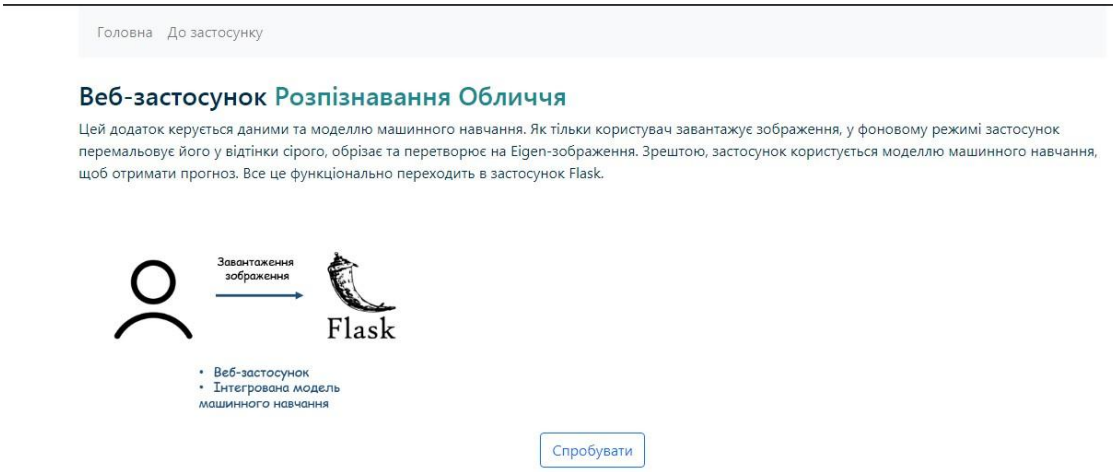

Рисунок 3.9 Сторінка переходу до програми

Далі, нам запропонують обрати файл з вашого персонального комп’ютеру, завантажити його та отримати прогноз:

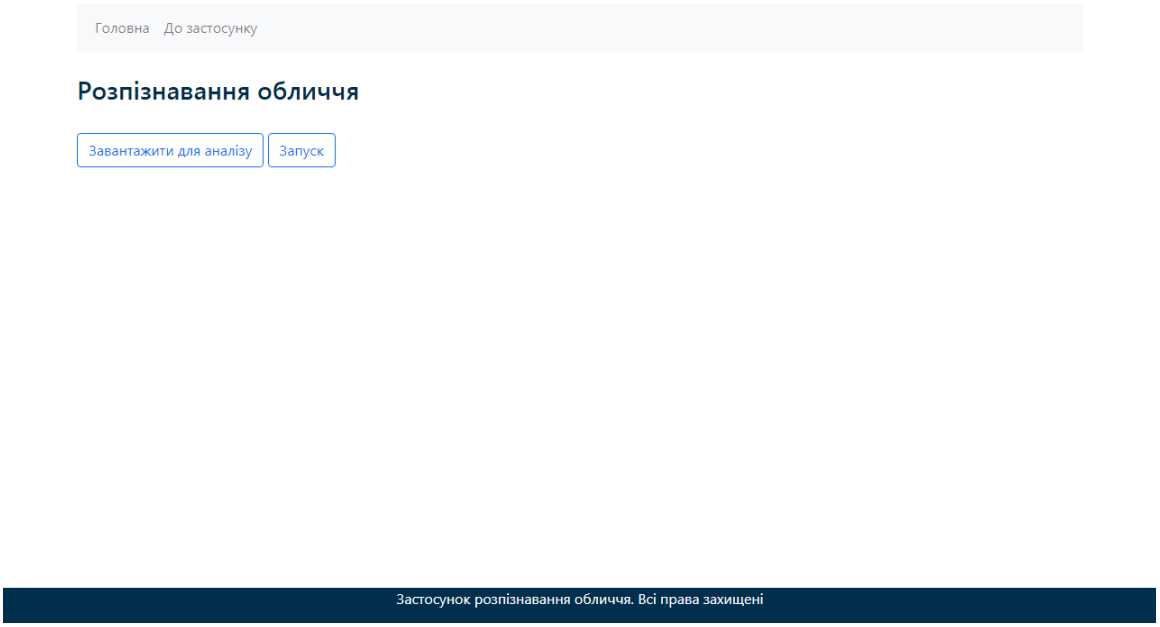

Рисунок 3.10 Завантаження файлу для розпізнавання

В результаті, ми отримуємо два зображення. Перше зображення являється початковим, тобто те, яке ви завантажили, а друге – результат роботи веб-застосунку з прогнозом.

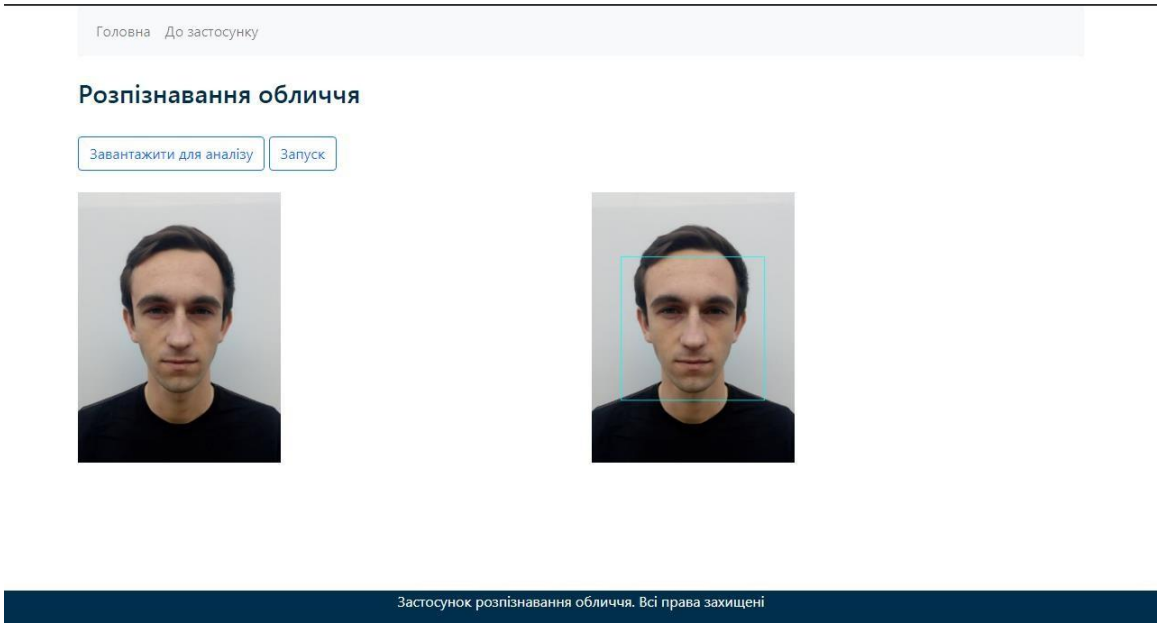

Рисунок 3.11 Результат виконання програми

Як бачимо, веб-застосунок розпізнав обличчя та вивів результат. Тепер трішки ускладнаємо задачу та спробуємо завантажити зображення з натовпом людей.

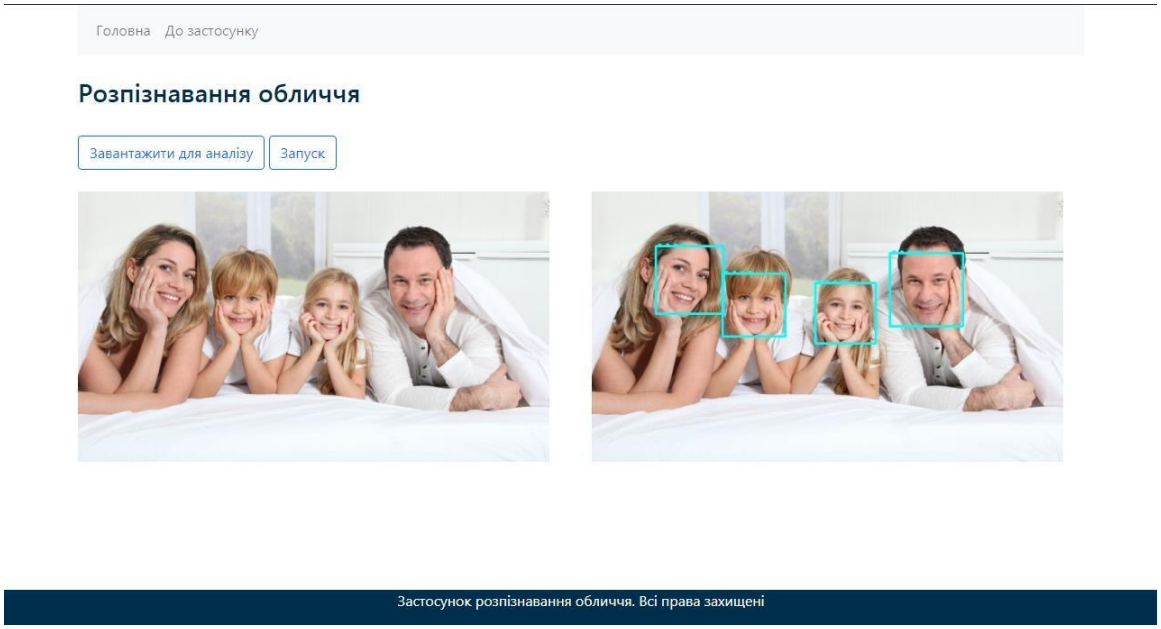

Рисунок 3.12 Приклад зображення з натовпом людей

Веб-застосунок справляється зі своєю роботою та розпізнає всі обличчя які знаходились на зображенні.

Для остаточної перевірки спробуємо завантажити зображення з наполовину обрізаним обличчям та обличчям, яке знаходиться поза зоною фокусу.

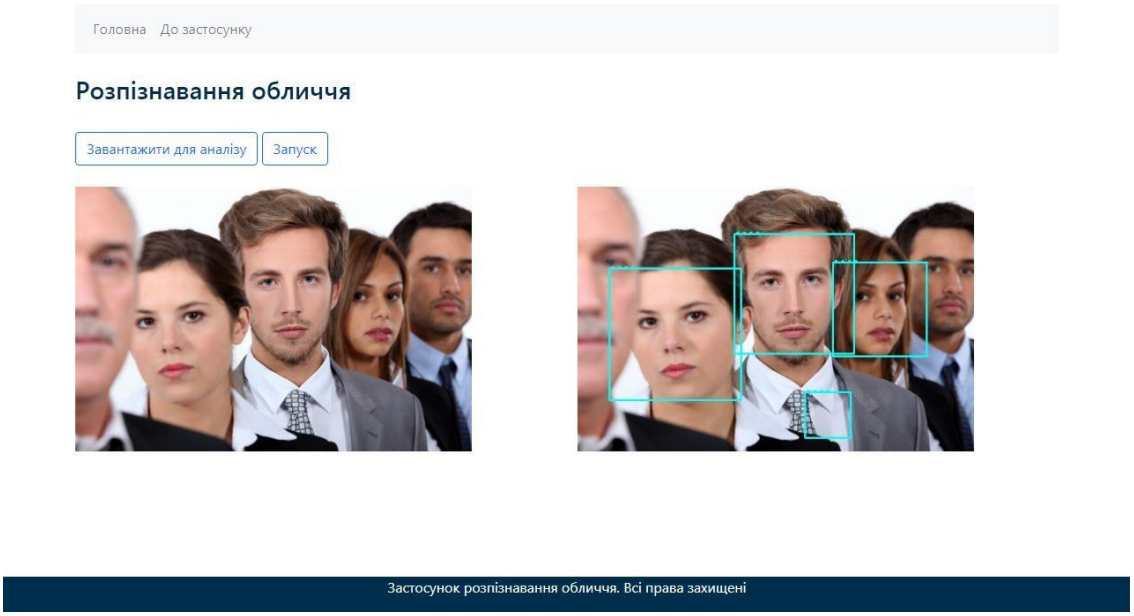

Рисунок 3.13 Приклад важкого для аналізу зображення

Висновки

У даному розділі мною висвітлено процес розроблення веб-застосунку розпізнавання обличчя.

Для проектування веб-застосунку було використано інструмент для інтерактивної розробки Jupyter notebook та розроблено модель машинного навчання. Для створення віртуального середовища та подальшого виконання файлів був залучений дистрибутив Anaconda Python.

В процесі розробки веб-застосунку було опрацьовано питання роутингу сторінок на принципі односторінкового застосунку. Обраними в розділі 2

інструментами розроблено сторінки веб-застосунку, створено їх функціонал, а самі сторінки об’єднано в логічно-працюючу схему.

Підбиваючи підсумки, можна сказати, що проектування та розробка веб-застосунку пройшли успішно, всі заявлені функції реалізовані, а використання програми відповідає сучасним вимогам та стандартам взаємодії з користувачем.

ВИСНОВКИ

В даній дипломній роботі було проаналізовано та розглянуто різноманітні способи та технології для створення веб-застосунку.

Досліджено які технології є більш зручними для створення веб-застосунку під різні потреби та функціонал. Також було наведено наочні та актуальні приклади найпопулярніших та розповсюджених інструментів для створення веб-сторінок.

Здійснено аналіз основних засобів побудови власної моделі машинного навчання та веб-застосунку розпізнавання обличчя.

Було реалізовано власний тип розпізнавання обличь, що дозволяє додавати до веб-застосунку зображення та аналізувати їх в режимі онлайн.

Робота виконана на мові програмування Python, яка має зв’язок з бібліотекою OpenCV та використовує плагіни, що було згадано раніше.

Після проведення тестування швидкості роботи та оптимізації веб-застосунку було здійснено тестування на працездатність в різноманітних веб-браузерах, таких як Google Chrome, Mozilla Firefox, Opera та інші. Також було проведено тестування працездатності на смартфонах, планшетах та ноутбуках. Всі модулі та блоки веб-застосунку знаходились на відповідних позиціях, та відображалися корректно.

СПИСОК ЛІТЕРАТУРИ

1. Гротер Р. Тест на розпізнавання обличчя (FRVT). Виконання алгоритмів ідентифікації обличчя. / Патрік Гротер, Мей Нган. - Відділ доступу до інформації Національний інститут стандартів і технологій. - 26 травня 2014 р. - р. 138.
2. Метод Віоли – Джонса. URL: https://habr.com/ru/post/133826/
3. Ірматов, А.А., Спосіб і система для розпізнавання особи з урахуванням списку людей, що не підлягають перевірці [Текст] / А. А. Ірматов, Д. Ю. Буряк - Корпорація «Самсунг електронікс Ко., Лтд.». - Москва: 2010. - 22 с.
4. Костецька, Г. Ю., Кодування зображень людських облич за допомогою самоорганізується карти Кохонена [Текст] / Г. Ю. Костецька, О. І. Федяєв - V міжнародна науково - технічна конференція студентів, аспірантів та молодих науковців «Інформатика та комп'ютерні технології »- Донецьк: ДонНТУ, 2009. - 268 с.
5. Технологія розпізнавання осіб / Data Систем. Товари та технології XXI. [Електронний ресурс]. - Режим доступу: http://hardbroker.ru/pages/recognition.
6. Горшенин, В. А., Геодезичний інваріант в біометрики особи [Текст] / В. А. Горшенин -. 2014. - 4 с. - (наук. Стаття)
7. Розпізнавання облич / Підкомітет NSTC з питань біометрії та управління особами. - 7 серпня 2006 р. - Режим доступу: http://www.biometrics.gov/documents/.
8. HTML5 Developer guides. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5.
9. Choose Between Traditional Web Apps and Single Page Apps (SPAs). Microsoft. URL: https://docs.microsoft.com/en-

us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps.

10. CSS: Cascading Style Sheets Developer guides. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/CSS.
11. Python (programming language) URL: https://en.wikipedia.org/wiki/Python_%28programming_language%29
12. Мальцев А. Використання каскаду Хаара для порівняння зображень / Мальцев Антон. - Режим доступу: https://habrahabr.ru/post/198338/
13. OpenCV крок за кроком. Інтегральне зображення. - Режим доступу: http://robocraft.ru/blog/computervision/53-6.html
14. Face Recognition / NSTC Subcommittee on Biometrics and Identity Management Room. - 7 August, 2006. - Режим доступу: http://www.biometrics.gov/documents/

Додатки

Лістинг основного файлу:

```
from flask import Flask
from app import views
app = Flask(__name__)

#url
app.add_url_rule('/base','base',views.base)
app.add_url_rule('/','index',views.index)
app.add_url_rule('/faceapp','faceapp',views.faceapp)
app.add_url_rule('/faceapp/app','app',views.app,methods=['GET','POST'])

#run
if__name___== "__main__":
    app.run(debug=True)
```

Лістинг налаштування рутів:

```
from flask import render_template, request
from flask import redirect, url_for
import os
from PIL import Image
from utils import pipeline_model

UPLOAD_FOLDER ='static/uploads'

def base():
    return render_template('base.html')
```

```
    def index():
        return render_template('index.html')

    def faceapp():
        return render_template('faceapp.html')

    def getwidth(path):
        img = Image.open(path)
        size = img.size # ширина й висота
        aspect = size[0] / size[1] # ширина / висота
        w = 300 * aspect
        return int(w)

    def app():
        if request.method =='POST':
            f = request.files['image']
            filename = f.filename
            path = os.path.join(UPLOAD_FOLDER,filename)
            f.save(path)
            # обробка
            w = getwidth(path)
            # передбачення
            img = pipeline_model(path,filename,color='bgr')
            return render_template('app.html',fileupload=True,img_name=filename,
w=w)

        return render_template('app.html',fileupload=False,img_name="test.png",w
 = 300)
```

Лістинг головної сторінки:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x" crossorigin="anonymous">
    <link rel="stylesheet" href="{{ url_for('static',filename='css/style.css') }}">
    <title>Розпізнавання обличчя</title>
</head>
<body>
    <div class="container">
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <div class="container-fluid">

                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
                    <div class="navbar-nav">
```

```
                    <a class="nav-item nav-
link" href="{{ url_for('index') }}">Головна</a>
                    <a class="nav-item nav-
link" href="{{ url_for('faceapp') }}">До застосунку</a>
                    </div>
                  </div>
                 </div>
               </nav>
          </div>

        {% block bodyblock %}



        {% endblock  %}

        <!--footer-->
        <footer class="footer">
          <div class="container">

            <p align="center">
              <span style="color:white">Застосунок розпізнавання обличчя. Всі пр
ава захищені</span>
            </p>

          </div>
        </footer>

      </body>
      </html>
```

Лістинг файлу роботи із зображеннями:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
import pickle
import cv2
import sys
sys.path.insert(1, './model/')

# завантаження моделі
haar = cv2.CascadeClassifier('./model/haarcascade_frontalface_default.xml')

# pickle-файли
mean = pickle.load(open('./model/mean_preprocess.pickle','rb'))
model_svm = pickle.load(open('./model/model_svm.pickle','rb'))
model_pca = pickle.load(open('./model/pca_50.pickle','rb'))

# налаштування
font = cv2.FONT_HERSHEY_SIMPLEX

def  pipeline_model(path,filename,color='bgr'):
    # зчитання зображення за допомогою cv2
    img = cv2.imread(path)
    # перетворення в сірий колір
    if color == 'bgr':
        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    else:
```

```
        gray = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
    # обрізання обличчя на зображенні
    faces = haar.detectMultiScale(gray,1.3,1)
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2) # малювання квадрат
y
        roi = gray[y:y+h,x:x+w] # обрізання зображення
        # нормалізація (0-1)
        roi = roi / 255.0
        # змінення розміру зображення (100,100)
        if roi.shape[1] > 100:
            roi_resize = cv2.resize(roi,(100,100),cv2.INTER_AREA)
        else:
            roi_resize = cv2.resize(roi,(100,100),cv2.INTER_CUBIC)

        roi_reshape = roi_resize.reshape(1,10000) #1,-1

        roi_mean = roi_reshape - mean

        text = '... '
        cv2.putText(img,text,(x,y),font,1,(255,255,0),2)

    cv2.imwrite('./static/predict/{}'.format(filename),img)
```