Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська дипломна робота на тему:

РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ДЛЯ РОЗПІЗНАВАННЯ ОБЛИЧ

Виконав: студент 4 курсу, групи 3КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Собщенко Р. О.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Крилик Л. В.
(прізвище та ініціали)
« 07 » червня 2024 р.

Рецензент: к.т.н., доцент каф. АІІТ
Богач І. В.
(прізвище та ініціали)
« 07 » червня 2024 р.

Допущено до захисту
Зав. кафедри Яровий А. А.
« 07 » червня 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
« 07 » 03.2024 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Собщенку Роману Олеговичу

1. Тема роботи: Розробка програмного модуля для розпізнавання облич.
керівник роботи: Крилик Людмила Вікторівна, к.т.н., доц.
затверджені наказом вищого навчального закладу « 11 » 03.2024 року № 80
2. Термін подання студентом роботи 27.05.2024 р.

3. Вихідні дані до роботи :
Мова програмування: об'єктно-орієнтована;
Формат вхідних даних: зображення або відеопотік;
Метод розпізнавання: використання нейронних мереж;
Кількість облич для розпізнавання одночасно: не менше 2 од.;
Кількість категорій для класифікації облич: не менше 2 од.;
Інтерфейс користувача: інтуїтивно зрозумілий і зручний для використання.

4. Зміст текстової частини (перелік питань, які потрібно розробити):
вступ; обґрунтування доцільності розробки програмного модуля для розпізнавання облич; проектування програмного модуля для розпізнавання облич; програмна реалізація програмного модуля для розпізнавання облич; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): схема алгоритму роботи програмного модуля для розпізнавання облич; загальна структурна схема функціонування системи; загальна UML-діаграма класів; структура принципу роботи програмного модуля для розпізнавання облич; UML-діаграма комунікації системи програмного модуля для розпізнавання облич; приклади роботи програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1–3	Крилик Л. В., к.т.н., доц. каф. КН		

7. Дата видачі завдання 07.03.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Вступ. Обґрунтування доцільності розробки програмного модуля для розпізнавання облич	06.05.2024	09.05.2024	
2	Розробка програмного модуля для розпізнавання облич	10.05.2024	14.05.2024	
3	Програмна реалізація модуля для розпізнавання облич	15.05.2024	21.05.2024	
4	Висновки та аналіз отриманих результатів	22.05.2024	24.05.2024	
5	Оформлення додатків	27.05.2024	29.05.2024	
6	Розробка інструкції користувача	30.05.2024	03.06.2024	
7	Оформлення матеріалів до захисту БДР	04.06.2024	06.06.2024	

Студент _____ **Собщенко Р. О.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Крилик Л. В.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 75 сторінок формату А4, які включають 30 рисунків і 4 таблиці. У списку використаних джерел зазначено 21 найменувань.

Основною метою роботи є розширення функціональних можливостей програмного модуля для розпізнавання облич. У загальній частині роботи на основі аналізу існуючих технічних рішень, методів та технологій доведена доцільність розробки програмного модуля для розпізнавання облич.

В роботі покращено алгоритм роботи програмного модуля для розпізнавання облич.

Для програмного модуля для розпізнавання облич на основі аналізу, обрана платформа OpenCV. Для кращої демонстрації роботи програми були розроблені такі діаграми: UML-діаграма класів та діаграма компонентів. Була створена схема алгоритму роботи модуля та загальна структурна схема функціонування системи. Результатом розробки є структура принципу роботи програмного модуля. Після проведення аналізу та тестування розробленого модуля були отримані результати, які вказують на ефективність та функціональність програми.

Розробка продукту була виконана в інтегрованому середовищі Visual Studio Code з використанням мови програмування Python.

Ключові слова: програмний модуль, розпізнавання облич, ініціалізація, алгоритм, проектування.

ABSTRACT

The bachelor's thesis consists of 75 A4 pages, which include 30 figures and 4 tables. The list of references contains 21 titles. The main objective of the work is to expand the functional capabilities of a face recognition software module. In the general part of the work, based on the analysis of existing technical solutions, methods, and technologies, the feasibility of developing a face recognition software module is proven.

The work improves the algorithm of the face recognition software module. Based on the analysis, the OpenCV platform was chosen for the face recognition software module. For better demonstration of the program's functionality, the following diagrams were developed: UML class diagram and component diagram. An algorithm workflow scheme and a general structural scheme of the system's functioning were created. The result of the development is the structure of the working principle of the software module. After conducting analysis and testing of the developed module, results were obtained that indicate the efficiency and functionality of the program.

The product development was carried out in the integrated environment of Visual Studio Code using the Python programming language.

Keywords: software module, face recognition, initialization, algorithm, design.

ЗМІСТ

ВСТУП

Актуальність теми.

Тема «Розробка програмного модуля розпізнавання облич» є надзвичайно актуальною в сучасному світі, де безпека, персоналізація та автоматизація процесів відіграють важливу роль. Технології розпізнавання облич широко використовуються в різних сферах, таких як безпека, маркетинг, охорона здоров'я та розваги. Програмні модулі, які базуються на розпізнаванні облич, можуть надати користувачам безліч переваг та покращити їхній досвід взаємодії з різними системами та сервісами.

Актуальність теми полягає в наступному:

- Підвищення рівня безпеки: технології розпізнавання облич використовуються для ідентифікації та верифікації особистості, це є важливим елементом у системах безпеки, зокрема, в доступі до будівель, аеропортів, банків та інших об'єктів.

- Персоналізовані послуги: розпізнавання облич дозволяє надавати користувачам персоналізовані послуги, адаптовані до їхніх індивідуальних потреб та вподобань, що є корисним у роздрібній торгівлі, маркетингу та обслуговуванні клієнтів.

- Автоматизація процесів: технології розпізнавання облич можуть автоматизувати різні процеси, такі як реєстрація користувачів, перевірка особистих даних та контроль доступу, це допомагає зменшити час та витрати на ці операції.

Загалом, розробка програмного модуля розпізнавання облич відповідає сучасним технологічним тенденціям, забезпечуючи безпечну, персоналізовану та ефективну підтримку різних процесів та послуг. Тому тема є надзвичайно актуальною і може бути цікавою для дослідження та розробки у рамках дипломної роботи.

Метою дослідження є розширення функціональних можливостей програмного модуля для розпізнавання облич.

Об’єктом дослідження є процес розпізнавання облич

Предметом дослідження є програмні засоби для розпізнавання облич.

Задачі дослідження:

1. Обґрунтувати доцільність розробки програмного модуля для розпізнавання облич;
2. Спроектувати програмний модуль для розпізнавання облич;
3. Програмно реалізувати модуль для розпізнавання облич;
4. Виконати тестування програми та провести аналіз отриманих результатів.
5. Розробити інструкцію користувача.

Апробація роботи.

Результати досліджень було апробовано на Міжнародній науково-практичній інтернет-конференції студентів аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2024) м. Вінниці у 2024 р. [1].

Публікації: за основними результатами досліджень опубліковано тези доповіді на науково-технічній конференції [1].

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ПРОГРАМНОГО МОДУЛЯ ДЛЯ РОЗПІЗНАВАННЯ ОБЛИЧ

1.1 Огляд середовищ для реалізації програмного продукту

За останні роки в галузі комп'ютерного зору виявлення та ідентифікація людей стали популярними напрямами досліджень, що є одними з найуспішніших застосувань алгоритмічного аналізу та розуміння зображень. Ключова проблема полягає в тому, щоб на вхідних нерухомих зображеннях чи відео виокремити та підрахувати одну або декілька осіб в приміщенні, використовуючи різні методи виявлення людей.

Виявлення осіб може здійснюватися двома способами:

- Виявлення обличчя − пошук обличчя на зображенні, відеопотоці чи в режимі реального часу.
- Виявлення тіла людини − пошук контурів людського тіла для локалізації та позначення його на зображенні, відеопотоці чи в реальному часі.

Хоча традиційні системи виявлення облич та людських фігур базуються переважно на нерухомих знімках, значний інтерес викликає розробка надійних систем, які можуть приймати відео на вхід. Відстеження людей набуло актуальності завдяки широкому розповсюдженню відеоспостереження. Можливість автоматично знаходити обличчя в режимі реального часу з відео допоможе ідентифікувати людей за допомогою існуючих камер відеонагляду. Проте виявлення облич на відео може ускладнюватися значними змінами освітлення, що погіршує продуктивність більшості систем.

Декілька кадрів забезпечують варіації поз людини, це дозволяє правильно вибрати якісний кадр (наприклад, чітке зображення обличчя) для підвищення ефективності виявлення. Тимчасова інформація у відео розглядається як дані, вбудовані в динамічний рух обличчя.

Система розпізнавання облич − це технологія, здатна перевіряти або ідентифікувати людей за цифровими зображеннями чи відеокадрами. Вона працює

шляхом порівняння характерних рис обличчя з наданого зображення з обличчями в базах даних. Такі системи засновані на біометричному штучному інтелекті та аналізують текстуру і форму людського обличчя для ідентифікації з певною ймовірністю [2].

Спочатку це були комп'ютерні програми, але нині вони широко використовуються на мобільних пристроях, у робототехніці та інших сферах застосування штучного інтелекту (ШІ). Найпоширеніше їх застосовують для контролю доступу в системах безпеки, порівнюючи з іншими біометричними показниками.

У 1997 році Крістоф фон дер Мальсбург та його студенти з університетів Бохума та Каліфорнії розробили передову систему розпізнавання облич на замовлення дослідницької лабораторії збройних сил США. Програмне забезпечення ZN-Face продавалося комерційним клієнтам і було досить надійним для ідентифікації за неідеальними зображеннями облич з перешкодами, як-от вуса, окуляри тощо.

Сучасні алгоритми в 10 разів точніші за старі 2002 року та в 100 разів кращі, ніж 10 років тому. Деякі здатні перевершити людей в ідентифікації і навіть розрізняти близнюків.

Основними проблемами є надлишковість пікселів зображення та різноманітність зовнішніх умов, за яких можуть бути зроблені фотографії (освітлення, фон, міміка тощо). Тому необхідно компактно подавати зображення для ефективної класифікації. Традиційні формати (JPEG, PNG) не дуже придатні для цього.

Розпізнавання облич ділиться на такі задачі: пошук в базах даних, контроль доступу та перевірка автентичності фотодокументів. Вони відрізняються вимогами та способами розв'язання. Є багато комерційних програм, як-от VOCORD FaceControl 3D і DeepFace, проте їхні алгоритми закриті [3].

Загалом, розробка якісних систем розпізнавання вимагає значних ресурсів і можлива лише для великих компаній та організацій. В основі лежить створення 3D-моделі обличчя та її порівняння з базою даних. Спрощені алгоритми працюють

так: від вхідного зображення відокремлюються риси обличчя, воно нормалізується до стандартного формату, після чого класифікатор визначає, чиє це обличчя або чи є обличчя взагалі.

На сьогодні існує багато платформ, на яких можна розробити такий програмний модуль. Щоб вибрати найкращий варіант, розглянемо і порівняємо популярні платформи, такі як OpenCV, Face++ та dlib [4]. Для більш зрозумілого порівняння, результати аналізу цих платформ наведені у таблиці 1.1:

Таблиця 1.1 – Порівняльна характеристика платформ для розробки програмного модуля

Платформа	***Точність***	***Інтеграція***	***Масштабованість***	***Переваги***	***Недоліки***
Microsoft Azure Face API	Висока	З іншими сервісами Azure	Висока	Простота, документація, підтримка мов	Вартість, потреба в інтернеті
Amazon Rekognition	Висока	З іншими сервісами AWS	Висока	Потужність, гнучкість, масштабованість	Вартість, потреба в інтернеті
Google Cloud Vision API	Висока	З іншими сервісами Google	Висока	Простота, потужність, підтримка мов	Вартість, потреба в інтернеті
OpenCV	Висока	Різні мови програмування	Середня	Безкоштовність, спільнота, локальна робота	Складність налаштування, відсутність підтримки
Face++ (Megvii)	Висока	Через API	Висока	Реальний час, документація, гнучкість	Вартість, потреба в інтернеті

OpenCV − це безкоштовна платформа з відкритим кодом для обробки зображень та машинного навчання, яка широко використовується для розробки програмного забезпечення для розпізнавання облич. Вона пропонує широкий спектр функцій для виділення ознак обличчя, навчання моделей машинного навчання та розпізнавання облич на нових зображеннях. OpenCV підтримує різні мови програмування, включаючи C++, Python та Java [5].

dlib − це ще одна безкоштовна платформа з відкритим кодом для обробки зображень та машинного навчання, яка часто використовується для розробки програмного забезпечення для розпізнавання облич, також пропонує подібний набір функцій, як і OpenCV, включаючи виділення ознак обличчя, навчання моделей машинного навчання та розпізнавання облич на нових зображеннях. dlib також підтримує різні мови програмування, включаючи C++, Python та Java.

Face++ − це комерційна платформа для розпізнавання облич, яка пропонує широкий спектр функцій. Вона включає в себе API для розпізнавання облич, відстеження облич, аналізу емоцій та інших завдань, пов'язаних з обробкою зображень. Face++ пропонує безкоштовний рівень обслуговування, а також платні плани з більшою кількістю функцій та обсягом обробки.

На основі аналізу даних з таблиці 1.1, стає очевидним, що найкращими варіантами для розробки програмного модуля є OpenCV та dlib. Проте, після уважного вибору віддаємо перевагу OpenCV. Це пояснюється тим, що OpenCV є більш функціональною та легко інтегруючою платформою для користувачів, в порівнянні з dlib та Face++.

1.2 Огляд модулів-аналогів розпізнавання облич

Існує кілька популярних і широко використовуваних модулів і бібліотек для розпізнавання облич, кожен з яких має свої унікальні особливості та переваги.

Багатозадачна каскадна згорткова нейронна мережа (MTCNN) − це сучасний потужний інструмент для виявлення облич на зображеннях, який використовує три послідовні етапи обробки за допомогою нейронних мереж (рис. 1.1) [6]. На першому етапі вхідне зображення масштабується декілька разів для виявлення облич різних розмірів. Потім мережа P-Net (Proposal Network) сканує зображення, виконуючи первинне виявлення облич. При цьому використовується низький поріг виявлення, щоб знайти максимальну кількість потенційних облич, через що виникає багато хибних спрацьовувань. Після застосування алгоритму видалення неосновних максимумів (NMS), виявлені області передаються другій мережі R-Net

(Refine Network), яка відфільтровує виявлення та уточнює обмежувальні рамки навколо облич. На завершальному етапі O-Net (Output Network) виконує фінальне вдосконалення обмежувальних рамок для отримання максимально точних результатів виявлення. Таким чином, MTCNN дозволяє не лише знаходити обличчя на зображеннях, а й забезпечує високу точність визначення їх розташування та обмежувальних рамок.

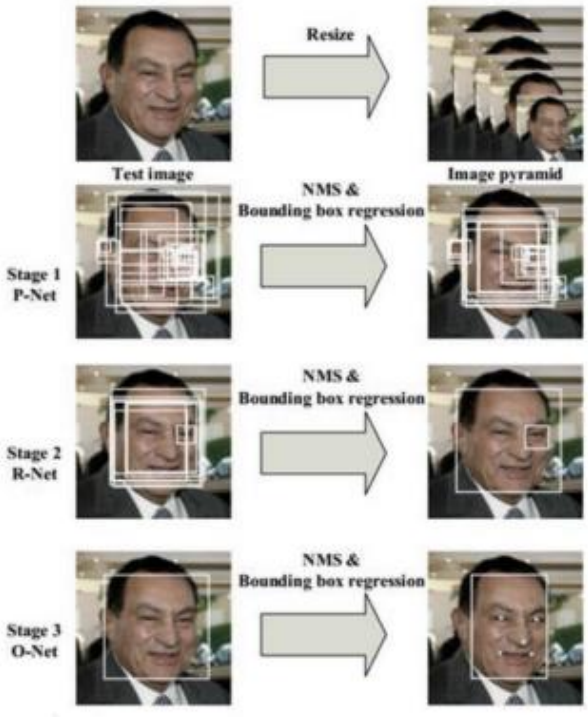

Рисунок 1.1 – Приклад роботи алгоритму MTCNN

MTCNN є дуже точним і надійним методом виявлення облич. Він належним чином розпізнає обличчя різних розмірів, при різному освітленні та значних поворотах. Хоча MTCNN працює дещо повільніше порівняно з детектором Віоли-Джонса, проте з використанням графічного процесора швидкодія не є критичною проблемою. Цей метод також використовує кольорову інформацію, оскільки згорткові нейронні мережі отримують зображення у форматі RGB на вході.

Існують результати дослідження щодо роботи різних алгоритмів (Haar Cascades, фронтальний детектор облич dlib, MTCNN та модуль DNN OpenCV) на зображеннях великих і малих розмірів:

- Haar Cascades є досить застарілим і демонструє загалом найгірші

результати.

- Модель розпізнавання облич модуля DNN OpenCV працює добре, але може мати проблеми з дуже великими зображеннями розміром 3000×3000 пікселів, що зазвичай не є типовим випадком.

- Dlib не розпізнає обличчя розміром менше 80×80 пікселів, тому при роботі з маленькими зображеннями потрібно їх масштабувати, що збільшує час обробки.

- MTCNN був би найкращим вибором, якщо потрібно мати справу з великими та малими обличчями, і натепер можна вважати його провідним методом виявлення облич.

Отже, MTCNN демонструє високу точність та надійність при виявленні облич різних розмірів, освітлення та орієнтації на даний момент.

LBPH простий в реалізації і ефективний метод розпізнавання облич (рис. 1.2) [7]. Основна ідея методу локальних бінарних шаблонів (Local Binary Patterns, LBP) полягає в узагальненні локальної структури зображення шляхом порівняння інтенсивності кожного пікселя з інтенсивністю його сусідніх пікселів. Для цього обирається центральний піксель і розглядається його оточення з сусідніх пікселів. Якщо інтенсивність центрального пікселя дорівнює або перевищує інтенсивність сусіднього пікселя, то для цього сусіднього пікселя присвоюється значення 1, інакше − значення 0.

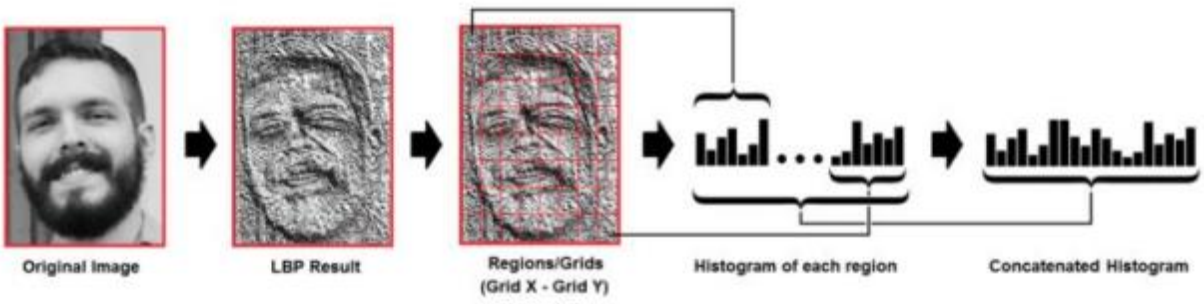

Рисунок 1.2 – Приклад роботи алгоритму LBPH

Таким чином, для кожного пікселя зображення формується бінарний код на основі порівняння його інтенсивності з інтенсивностями пікселів у його

локальному оточенні. Цей бінарний код описує локальну текстурну структуру навколо розглянутого пікселя і може бути використаний для подальшого аналізу та розпізнавання текстур і об'єктів на зображенні.

У випадку, коли точка даних потрапляє між піксельними координатами, застосовується білінійна інтерполяція. Для обчислення значення цієї нової точки використовуються значення 4 найближчих пікселів, що її оточують, утворюючи квадрат 2×2. Таким чином, шляхом зваженого усереднення значень сусідніх пікселів, визначається апроксимоване значення для проміжної точки, яка не збігається з дискретними піксельними координатами.

Іншими словами, якщо необхідно отримати значення в точці, яке не співпадає з центрами пікселів зображення, то білінійна інтерполяція дозволяє обчислити наближене значення в цій точці на основі зважених значень 4 найближчих до неї пікселів, розташованих у вузлах квадрата 2×2 навколо шуканої точки на зображенні.

Його основні переваги включають:

- LBPH − один з найпростіших алгоритмів розпізнавання облич.
- Він може представляти місцеві особливості на зображеннях.
- Можна отримати гарні результати (переважно в контрольованому середовищі).
- Він стійкий до монотонних перетворень шкали сірого.
- Він надається бібліотекою OpenCV [8].

CNN − нині існує багато різновидів нейронних мереж. Їхньою основною особливістю є здатність до навчання. Навчання нейронної мережі відбувається на готових прикладах. Під час навчання виділяються ключові ознаки, і встановлюються взаємозв'язки між ними. Після навчання нейронна мережа може застосовувати набутий досвід для розпізнавання раніше невідомих об'єктів.

Згорткові нейронні мережі (CNN) демонструють найкращі результати в задачах розпізнавання, проте вважаються найскладнішими для реалізації. CNN можуть враховувати двовимірну топологію зображень. Основні особливості CNN: загальні ваги (можливість виявляти об'єкти в будь-якій частині зображення),

локальні рецептивні поля (ділянки з рецепторами двовимірних зв'язаних нейронів). Ці особливості забезпечують стійкість до різноманітних спотворень (зсуву, змін масштабу тощо).

Отже, нейронні мережі мають унікальну здатність до навчання на прикладах для подальшого розпізнавання невідомих об'єктів. Згорткові нейронні мережі є потужним інструментом для розпізнавання об'єктів на зображеннях завдяки врахуванню їх двовимірної структури та стійкості до спотворень. Проте їх складність робить реалізацію доволі складним завданням (рис. 1.3) [9].

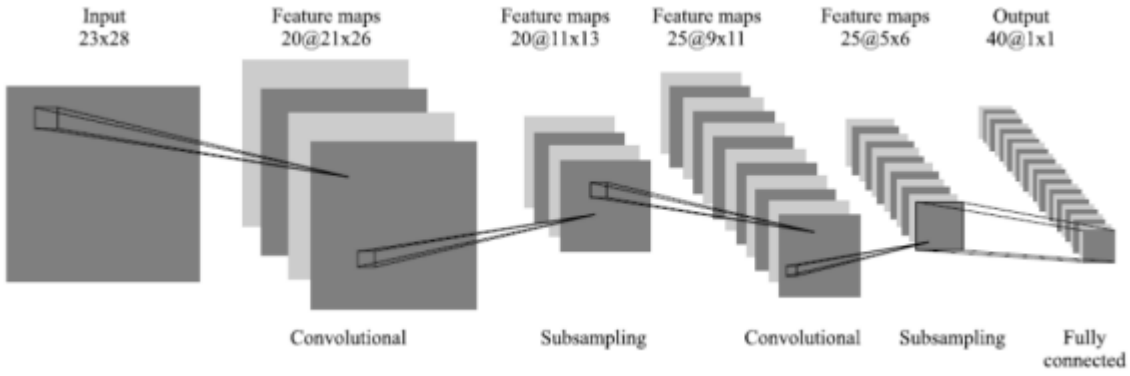

Рисунок 1.3 – Приклад архітектури CNN

Основні недоліки методу:

- важка реалізація (велика кількість та складність шарів), перенавчання при додаванні еталонного обличчя в базу даних [10].

FaceNet − це нейронна мережа, яка навчається відображати зображення облич у компактний евклідовий простір, де відстань між точками відповідає мірі схожості облич. Архітектура мережі складається з вхідного пакетного шару та глибокої згорткової нейронної мережі CNN, за якою йде нормалізація L2 для отримання дескриптора обличчя. Під час навчання використовується особлива функція втрат − TripletLoss.

TripletLoss мінімізує відстань між якірним зображенням обличчя та зображеннями тієї самої особи (позитивними прикладами), водночас максимізуючи відстань між якірним зображенням та зображеннями інших осіб (негативними прикладами). Таким чином, FaceNet навчається відображати схожі обличчя

близько одне до одного в евклідовому просторі, а несхожі - на значній відстані.

Отже, FaceNet є потужною нейронною мережею для задач розпізнавання облич, яка кодує зображення обличчя у векторні дескриптори в компактному просторі, де схожі обличчя знаходяться поблизу, а несхожі - віддалені одне від одного. Це досягається завдяки спеціальній функції втрат TripletLoss під час навчання мережі (рис. 1.4) [11].

Рисунок 1.4 – Структура моделі FaceNet

Однією з найважливіших переваг у FaceNet − це вибір триплетів для навчання. Триплети потрібно добирати таким чином, щоб для якірного зображення x^a вибрати як "найскладніший позитивний" приклад (зображення тієї самої особи) x^p той, що знаходиться на достатній відстані в наборі даних від якоря, а як "найскладніший негативний" приклад (зображення іншої особи) x^n − той, що розміщений найближче до якоря. Якщо такий триплет не порушує умов навчання, то жоден інший триплет з цим якорем не буде кориснішим.

Крім того, FaceNet є сіамською нейронною мережею. Сіамські мережі − це особлива архітектура, яка навчається диференціювати вхідні дані, тобто розрізняти схожі та несхожі приклади. Головною перевагою FaceNet є висока репрезентативна ефективність − можливість досягати найсучасніших показників (рекорд 99,63% на LFW, 95,12% на Youtube Faces DB). Мережа вміє ідентифікувати схожі обличчя при різних виразах та ракурсах, а також працювати з різним освітленням та перетвореннями зображень.

Отже, ретельний вибір складних навчальних триплетів та використання сіамської архітектури дозволяє FaceNet навчатися ефективно відображати схожі обличчя близько одне до одного у векторному просторі, демонструючи найвищі

результати на тестових наборах даних облич.

ResNet − архітектура залишкових згорткових мереж (ResNet) була розроблена для вирішення проблеми зникаючого градієнта при тренуванні дуже глибоких нейронних мереж. Традиційні моделі конволюційних нейронних мереж можуть ставати менш ефективними при додаванні нових шарів через насичення або погіршення продуктивності внаслідок зникаючого градієнта.

Проблема зникаючого градієнта виникає через процес оптимізації вагових коефіцієнтів методом зворотного поширення градієнта. Для дуже глибоких мереж зі значною кількістю шарів багаторазове множення градієнтів призводить до того, що градієнт "зникає" або прямує до нуля, не дозволяючи ефективно оновлювати ваги.

Архітектура ResNet пропонує інноваційний підхід для вирішення цієї проблеми. Завдяки залишковим зв'язкам між шарами та обчисленням залишкового градієнта можливо навчати глибокі згорткові нейронні мережі з сотнями і тисячами параметрів, уникаючи проблем зникаючого градієнта. Саме переваги архітектури ResNet − ефективне тренування дуже глибоких моделей із численними параметрами роблять її ідеальним рішенням для складних задач, зокрема в комп'ютерному зорі (рис. 1.5).

Основні переваги методу:

- ResNet відносно легко оптимізувати: «прості» мережі (які просто складають шари) показують велику помилку навчання, коли глибина збільшується.

- ResNet дозволяє відносно легко збільшити точність завдяки збільшенню глибини, чого з іншими мережами домогтися складніше.

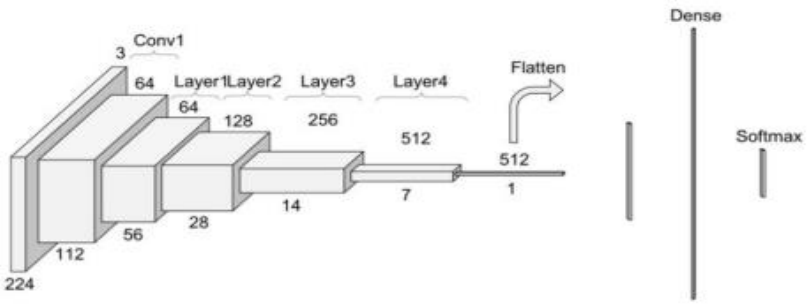

Рисунок 1.5 – Приклад архітектури ResNet-34

Розглянувши алгоритми з розпізнавання облич бачимо, що ResNet, CNN, FaceNet, LBPH та MTCNN − це надійні та широко використовувані методи для ідентифікації облич та отримання інформації про їх особливості. ResNet − це потужна нейронна мережа, яка показує високі результати в різноманітних задачах розпізнавання. CNN (Convolutional Neural Network) − класична архітектура для обробки зображень, що добре справляється з класифікацією та розпізнаванням облич. FaceNet − метод, який використовує глибокі нейронні мережі для отримання компактних векторних представлень облич (ембедінгів), що дозволяє ефективно порівнювати обличчя. LBPH (Local Binary Patterns Histograms) − підхід, що базується на текстурних ознаках зображення і є менш вимогливим до ресурсів. MTCNN (Multi-task Cascaded Convolutional Networks) − комплексна система, що одночасно виконує детекцію облич, вирівнювання та витягування ключових точок.

Кожен з цих методів має свої переваги та може бути корисним у відповідних ситуаціях. Виходячи з вище перерахованого, бачимо перспективи для розвитку та покращення, розроблюваного методу у майбутньому.

1.3 Формулювання вимог та постановка задачі

В нашому сучасному світі тема програмного модуля розпізнавання облич є дуже актуальною. Завдяки швидкому розвитку технологій та поширенню камер і систем відеоспостереження, люди все більше використовують ці технології для безпеки, ідентифікації та персоналізації. Програмний модуль для розпізнавання облич може забезпечити користувачам зручний та ефективний інструмент, надаючи можливість ідентифікувати особи, здійснювати контроль доступу, а також забезпечувати безпеку та аналітику.

Такий модуль може надавати високоточне розпізнавання, відповідати на запити щодо ідентифікації, допомагати з верифікацією осіб та багатьма іншими питаннями, що стосуються безпеки та управління доступом. Він може бути цінним помічником для підприємств, державних установ та приватних осіб, допомагаючи зекономити час, підвищити рівень безпеки та отримати персоналізовані дані. Він

також може підвищити рівень зручності та ефективності, допомагаючи знайти потрібну інформацію в режимі реального часу.

Програмний модуль розпізнавання облич матиме такі функції:

- визначення осіб за допомогою різних алгоритмів, таких як ResNet, CNN, FaceNet, LBPH та MTCNN;
- ідентифікація та верифікація осіб у реальному часі;
- надання інформації про точність та надійність розпізнавання;
- збереження та управління даними про ідентифіковані обличчя.

У цій роботі потрібно створити програмний модуль розпізнавання облич.

Модуль буде використовуватись для автоматичного визначення осіб (співробітники, відвідувачі, клієнти тощо). Модуль має бути максимально зручним для користувачів і розроблятися з урахуванням користувацького досвіду (UX). Вхідні дані, такі як зображення та налаштування мають легко задаватися користувачем шляхом інтуїтивного та зрозумілого інтерфейсу. Вихідні дані мають бути чітко й зрозуміло представлені, щоб користувач міг легко отримати потрібну інформацію.

В розроблюваному модулі для розпізнавання облич, зручність є одним із ключових аспектів. Модуль має забезпечувати користувачу можливість легко та швидко знаходити потрібну інформацію. Він має надавати доступ до актуальних даних про розпізнані обличчя, вказувати точність розпізнавання та надавати інформацію про ідентифікованих осіб. Програма матиме інтуїтивно зрозумілий та простий інтерфейс, щоб користувачі могли швидко зорієнтуватися та отримати потрібну інформацію у будь-який момент.

Для написання коректного інтуїтивного модуля для розпізнавання облич, потрібно вирішити такі завдання:

1. Розробити програмний модуль покрокового супроводу користувача для розпізнавання облич;

2. Реалізувати функцію легкого та швидкого знаходження актуальних даних про осіб;

3. Реалізувати функцію ідентифікації та верифікації осіб у реальному часі;

4. Реалізувати функцію надання інформації про точність та надійність розпізнавання;

5. Здійснити тестування програмного модуля розпізнавання облич.

1.4 Висновок

У цьому розділі було проведено аналіз існуючих платформ для розробки модулів розпізнавання облич і LBPH було обрано як найбільш прийнятну через свою потужність та доступну документацію. Були розглянуті існуючі рішення і виявлено, що вони не повністю задовольняють потреби користувачів і нерідко мають обмежену функціональність або складність у налаштуванні.

Тема програмного модуля розпізнавання облич є актуальною в наш час, оскільки технології безпеки та ідентифікації осіб стають все більш необхідними. Програмні модулі розпізнавання облич забезпечують швидкий доступ до надійних методів ідентифікації, верифікації, контролю доступу та аналізу, полегшуючи процес забезпечення безпеки та управління даними про осіб.

2 РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ДЛЯ РОЗПІЗНАВАННЯ ОБЛИЧ

2.1 Розробка структури програмного модуля розпізнавання облич

Подібно до програм, програмний модуль розпізнавання облич складається з набору функціональних компонентів, але зберігати всі ці компоненти в одному файлі незручно і непрактично. Структура такого модуля ґрунтується на розбитті на окремі частини або модулі. Кожен модуль відповідає за конкретну афункціональність або частину модуля розпізнавання облич. Наприклад, можуть бути окремі модулі для детекції облич, їх вирівнювання, витягнення ознак, класифікації та збереження результатів.

Кожен модуль програмується та розробляється незалежно від інших модулів, це дозволяє працювати паралельно та зосередитись на конкретній функціональності без необхідності змінювати весь код модуля розпізнавання облич. Крім того, така структура дозволяє зручно підтримувати та масштабувати програмний модуль у майбутньому.

Кожен модуль має свою назву та може мати свою власну структуру файлів, що спрощує розуміння та організацію коду. Наприклад, модуль може містити окремі файли для моделей машинного навчання, скриптів обробки зображень, конфігураційних файлів та результатів тестування. Застосування модульної структури дозволяє зберігати код програмного модуля організованим, зрозумілим та легко розширюваним.

В програмному модулі розпізнавання облич будуть такі компоненти: модуль для детекції облич, модуль для вирівнювання облич, модуль для витягнення ознак, модуль для класифікації облич та модуль для збереження результатів. Кожен компонент матиме свої підмодулі: модуль для графічного відображення результатів, для обробки даних, для реалізації функціональності клієнтської частини та для реалізації серверної частини, яка буде взаємодіяти з базою даних. Структура програмного модуля розпізнавання облич зображена на (рис. 2.1).

На етапі детекції облич зображення обробляється для виявлення облич, після чого ці дані передаються на наступний етап за допомогою модуля для реалізації функціональності клієнтської частини, після чого оброблюються на сервері.

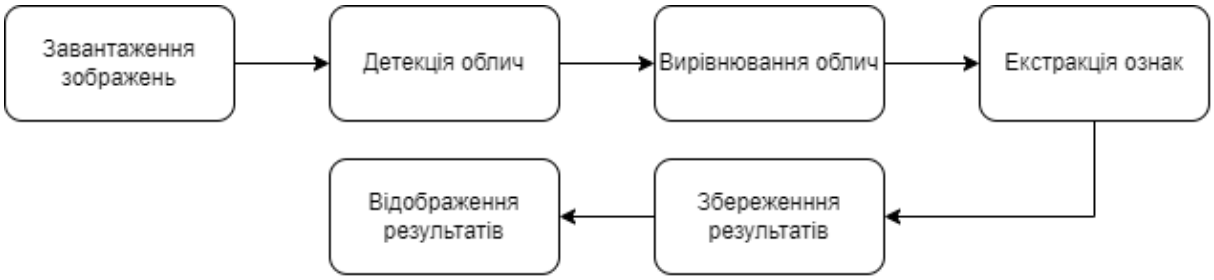

Рисунок 2.1 − Загальна структурна схема функціонування системи

На вхідних зображеннях обличчя представлені окремо. Однак виникає проблема, коли обличчя повернуті в різні сторони, вони виглядають абсолютно по-різному для комп'ютера.

Щоб врахувати це, необхідно викривити кожне зображення таким чином, щоб очі та губи завжди займали стандартні позиції. Це значно спростить порівняння облич на наступних етапах.

Для цієї мети використовується алгоритм, який називається "оцінкою орієнтирів обличчя". Існує багато способів його реалізації, але в такому випадку буде застосовано алгоритм, запропонований у 2014 році Вахідом Каземи та Джозефін Галліван.

Основна ідея полягає у виділенні 68 специфічних точок (званих орієнтирами), які присутні на кожному обличчі - верхня точка підборіддя, зовнішні краї очей, внутрішні краї брів тощо. Потім алгоритм машинного навчання налаштовується на знаходження цих 68 конкретних точок-орієнтирів на будь-якому обличчі.

Таким чином, за допомогою оцінки орієнтирів обличчя виконується його вирівнювання для подальшої ефективної обробки.

Структура та загальний принцип роботи програмного модуля розпізнавання облич подано на рис. 2.2.

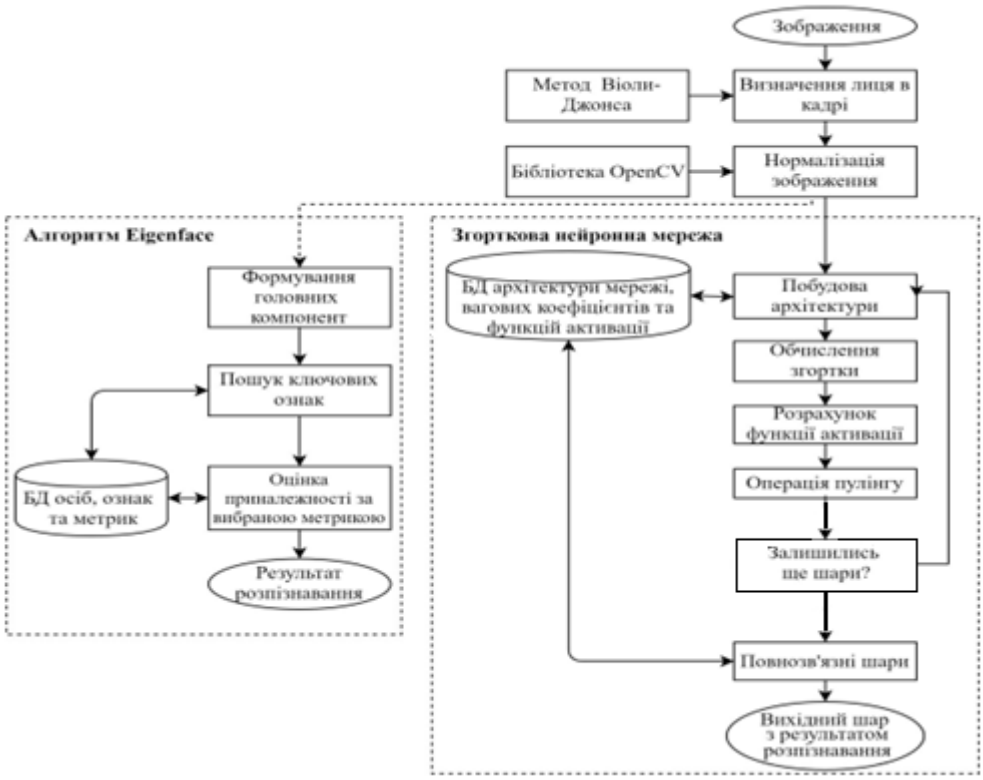

Рисунок 2.2 – Структурна схема принципу роботи модуля розпізнавання облич

2.2 Розробка UML-діаграми класів та комунікацій

UML діаграма класів (Class diagram) - це графічне представлення моделювання структури системи, яка показує класи, їх характеристики, функції та взаємозв'язки між ними. Вона використовується для візуалізації статичних аспектів системи, таких як класи, їх властивості та відношення (рис. 2.3) [10].

Умовні позначення на UML діаграмі класів:

1. Класи відображаються у прямокутних фігурах, які містять назву класу. Риси класу відображаються у вигляді змінних зі знаком «+» або «-» для вказання видимості;

2. Операції класу відображаються у вигляді функцій зі знаком «+» або «-» для вказання видимості;

3. Зв'язки між класами показуються за допомогою стрілок. Різні типи зв'язків включають відношення успадкування (inheritance), асоціації (association), агрегації (aggregation) та композиції (composition).

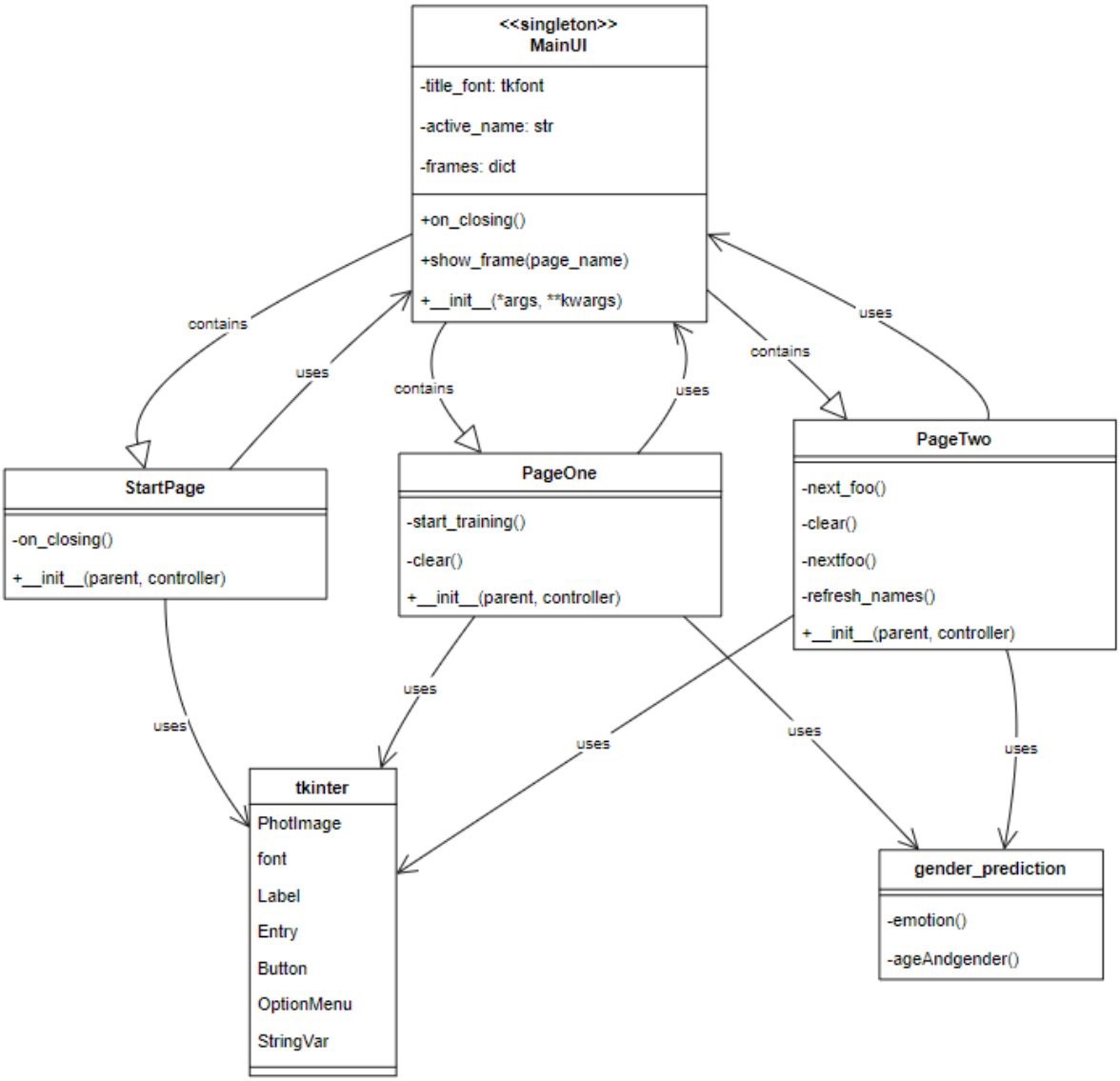

Рисунок 2.3 – Загальна UML-діаграма класів програмного модуля для розпізнавання облич

UML-діаграма комунікації є одним з типів діаграм UML і використовується для візуалізації взаємодії між об'єктами або компонентами системи. Вона показує, як об'єкти взаємодіють між собою за допомогою повідомлень або повідомлень з використанням послідовності часу [11].

У випадку програмного модуля розпізнавання облич, UML-діаграма комунікації може показати взаємодію між різними модулями системи або між системою та користувачем. Наприклад, діаграма може відображати взаємодію між модулем графічного інтерфейсу і модулем обробки зображень, де графічний

інтерфейс може надсилати запити на розпізнавання облич, а модуль обробки зображень може відповідати на ці запити, надсилаючи результати розпізнавання.

Діаграма комунікації також може показати обмін повідомленнями між користувачем та системою через інтерфейс або мобільний додаток. Вона може показувати, як користувач завантажує зображення до системи, як система обробляє це зображення та надсилає результат розпізнавання користувачу.

У діаграмі комунікації можуть бути зображені об'єкти, повідомлення, лінії життя (lifeline), сигнали та інші елементи, які відображають взаємодію між об'єктами або компонентами системи.

Використання UML-діаграми комунікації допомагає зрозуміти взаємодію між різними частинами системи, ідентифікувати послідовність повідомлень і розробити більш чітке розуміння функціональності системи (рис. 2.4).

Рисунок 2.4 – UML-діаграма комунікації системи програмного модуля для розпізнавання облич

2.3 Розробка схеми алгоритму роботи програмного модуля розпізнавання облич

Алгоритм − це послідовний набір вказівок, які призначені для вирішення певної задачі і завжди мають початок і кінець. Якщо алгоритм не має кінцевої точки, то під час розробки сталася помилка. Використання алгоритму у системі або додатку спрощує процес розробки, зокрема логіку в конкретному модулі.

Алгоритм можна описати двома способами:

- Лінгвістичний − це опис алгоритму за допомогою слів. Головним недоліком цього підходу є те, що словесний опис може бути забутий, і розробникові доведеться повторно уточнювати всі деталі;
- Схематичний − це опис алгоритму за допомогою структурних блоків. Великою перевагою цього підходу є те, що потрібно створити схему лише один раз і передати її програмісту.

Отже, для розробки програмного модуля розпізнавання облич оберемо схематичний спосіб опису. Алгоритм роботи цього модуля наведено на рисунку 2.5.

І випадок

Алгоритм у випадку розпізнавання обличчя в реальному часі складається з таких кроків:

Крок 1. Користувач запускає програму.

Крок 2. Програма активує WEB-камеру;

Крок 3. Програма захоплює кадри з WEB-камери;

Крок 4. Програма обробляє кожен кадр для виявлення облич;

Крок 5. Програма порівнює виявлені обличчя з базою даних відомих облич;

Крок 6. Програма відображає оброблені кадри користувачу в реальному часі;

Крок 7. Виведення результату.

ІІ випадок

Алгоритм у випадку додавання нового обличчя до бази даних складається з таких кроків:

Крок 1. Користувач запускає програму;

Крок 2. Програма активує WEB-камеру;

Крок 8. Програма захоплює кадри з WEB-камери;

Крок 9. Користувач вводить ім'я для нового обличчя;

Крок 10. Програма виявляє обличчя на зображенні;

Крок 11. Програма зберігає обличчя та ім'я в базу даних відомих облич;

Крок 12. Програма підтверджує додавання нового обличчя користувачу;

Крок 7. Виведення результату.

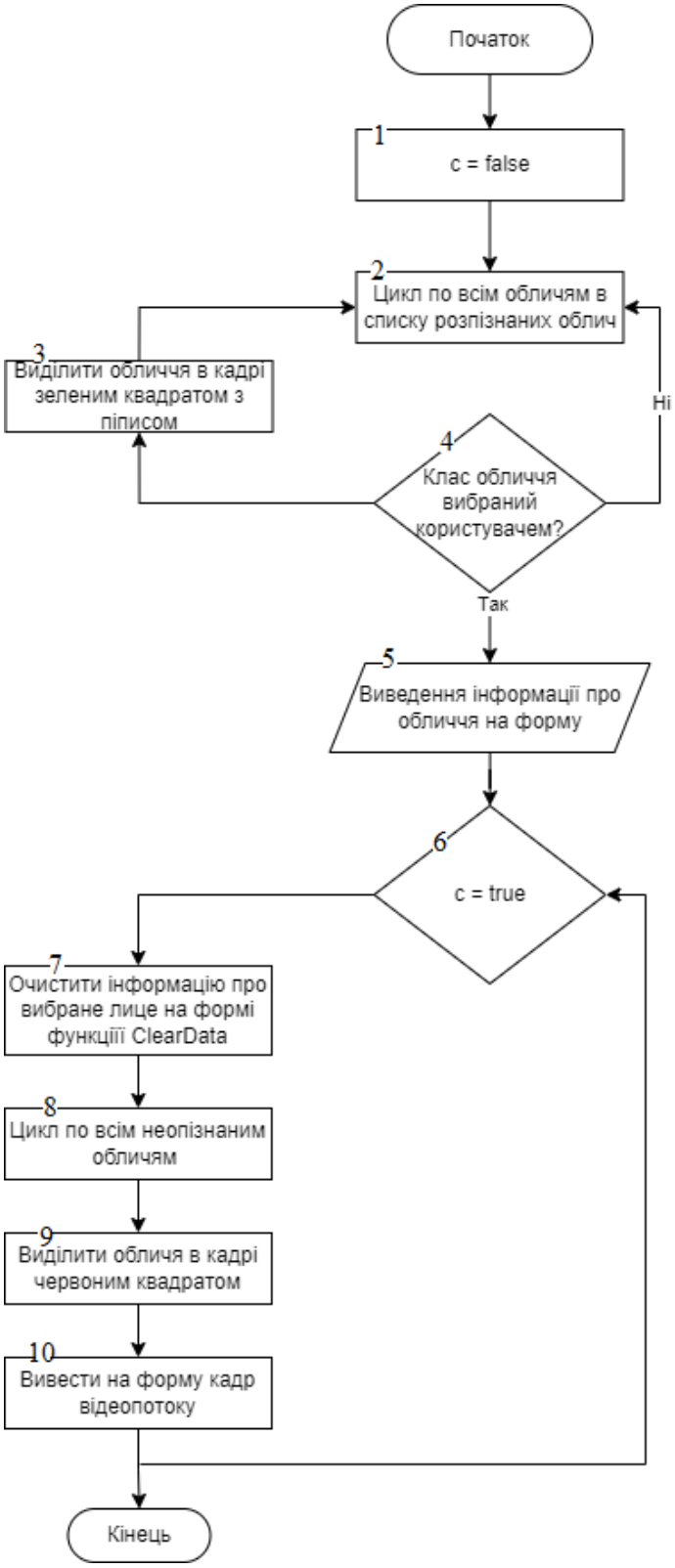

Рисунок 2.5 – Схема алгоритму роботи програмного модуля для розпізнавання облич

III випадок

Алгоритм у випадку видалення обличчя з бази даних складається з таких кроків:

Крок 1. Користувач запускає програму;

Крок 13. Користувач натискає кнопку "Видалити обличчя";

Крок 14. Програма відображає список усіх відомих облич;

Крок 15. Користувач вибирає обличчя, яке потрібно видалити;

Крок 16. Користувач підтверджує видалення;

Крок 17. Програма видаляє вибране обличчя з бази даних;

Крок 18. Користувач вибирає обличчя, яке потрібно видалити;

Крок 19. Програма підтверджує видалення користувачу;

Крок 7. Виведення результату.

2.4 Висновок

У цьому розділі були проаналізовані та сформульовані основні вимоги до програмного та апаратного забезпечення для розробки програмного модуля розпізнавання облич.

Була розроблена UML-діаграма комунікації системи розпізнавання облич, яка використовується для візуалізації та моделювання взаємодії між об'єктами або компонентами в системі. Вона дозволяє представити, як об'єкти спілкуються один з одним та передають повідомлення під час виконання конкретного сценарію або функції.

Для подальшого уточнення розробки була побудована загальна UML-діаграма класів, яка включає опис методів цих класів. Візуалізація коду у вигляді діаграм класу допомагає швидше отримати інформацію про структуру класу та його методи, а також про поведінку об'єктів. Після проектування діаграми класу розробник краще розуміє предметну галузь та може зробити декомпозицію, аналізуючи сутність для якої проектується клас і визначити, що потрібно врахувати, а що можна проігнорувати.

Крім того, була розроблена схема алгоритму функціонування програмного модуля для розпізнавання облич. Схема алгоритму допомагає зрозуміти, як покроково буде працювати програма. Вона дозволяє програмісту краще розуміти поставлену задачу та запобігає можливості неправильного функціонування деяких функціональних елементів. Всі ці кроки сприяють ефективній розробці програмного модуля розпізнавання облич, забезпечують чітке розуміння комунікації системи розпізнавання, структури, методів класів та алгоритму функціонування, що є ключовим для успішної реалізації розробки.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ РОЗПІЗНАВАННЯ ОБЛИЧ

3.1 Обґрунтування вибору мови та бібліотек програмування

Розуміючи всю важливість розпізнавання облич, усвідомлюємо актуальність цієї теми для багатьох сфер життя. Тому було прийнято рішення реалізувати програмний модуль для розпізнавання облич за допомогою мови програмування Python.

Python є мовою програмування, створеною розробником Гвідо ван Россумом у 1990 році, яка має простий синтаксис та високу динамічну типізацію. Використання Python можна зустріти в системному адмініструванні, наукових дослідженнях, WEB-розробках та мобільних додатках.

Особливості мови Python включають:

1. Простота: завдяки легкості, простоті та лаконічності, мова програмування є гарним вибором для початківців у програмуванні;

2. Багатство бібліотек: Python надає великий вибір вбудованих модулів і функцій, що дозволяє користувачам та розробникам швидко вирішувати завдання для реалізації своїх ідей без необхідності сторонніх бібліотек;

3. Підтримка різних парадигм програмування: Python підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє розробникам використовувати різні підходи для розробки;

4. Портативність: для запуску програми у багатьох мовах програмування потрібно вносити зміни у коді, але не у Python. Вона запам'ятовує один раз і може запускатись на будь-яких пристроях без додаткових керувань.

Недоліками мови програмування Python є:

- Швидкість: Python є інтерпретованою мовою без виконання попередньої компіляції до машинного коду, тому швидкість суттєво зменшується у порівнянні з мовами програмування C++, Objective-C тощо;

- Динамічна типізація: Python має динамічну типізацію, це дозволяє змінювати типи даних та змінювати опис узагальнених алгоритмів під час виконання програми.

- C# – мова програмування, яка використовується для розробки додатків під платформу .NET. Ця мова програмування дозволяє створювати додатки з багатим функціоналом та високою продуктивністю.

Основними перевагами C# є:

1. Широке використання: C# використовується для розробки різноманітних додатків, включаючи WEB-додатки, ігри, десктопні програми та мобільні додатки. Завдяки таким можливостям розробники можуть створювати багатофункціональні додатки на одній мові програмування;

2. Статична типізація: типи змінних у C# встановлюються під час компіляції, це дозволяє уникнути багатьох помилок, які можуть виникнути під час виконання програми;

3. Продуктивність: C# має високу продуктивність завдяки компіляції до проміжного коду, який потім виконується віртуальною машиною .NET, що дозволяє ефективно використовувати ресурси пристрою;

4. Інтеграція з платформою .NET: C# забезпечує тісну інтеграцію з платформою .NET, що дозволяє розробникам використовувати потужні інструменти та бібліотеки для швидкої і ефективної розробки додатків.

Основні недоліки C# включають:

- Залежність від платформи: C# в основному використовується для розробки додатків під платформу Windows, хоча останнім часом з'явилися можливості для кросплатформеної розробки за допомогою .NET Core;

- Складність: C# є більш складною мовою програмування у порівнянні з Python, що може бути викликом для початківців;

- Швидкість розробки: завдяки статичній типізації та необхідності компіляції, швидкість розробки може бути меншою у порівнянні з Python.

Проведемо коротку порівняльну характеристику між C# та Python за такими категоріями:

- Синтаксис: Python має простий та зрозумілий синтаксис, який робить цю мову особливо зручною для початківців. C# відрізняється складністю, але надає більше можливостей для оптимізації та контролю;
- Сфера застосування: зазвичай C# використовується для розробки десктопних і WEB-додатків під платформу Windows, ігор та мобільних додатків. Python має широке застосування у сфері науки, машинного навчання та обробки даних.

У таблиці 3.1 подана порівняльна характеристика мов програмування

Таблиця 3.1 – Порівняльна характеристика мов програмування

Назва критерію	C#	Python	Java
Динамічна типізація	+	-	+
Неявне приведення до типів	+	-	+
Псевдоніми типів	+	+	-
Виведення сигнатури для локальних методів	-	+	-
Багатопоточна компіляція	-	+	+
Створення об’єктів у стеку	+	+	-
Спискові виключень	+	+	-
Значення параметрів за замовчуванням	+	-	+

Отже, використання мови програмування Python для розробки модуля для розпізнавання облич має переваги у простоті вивчення та читання коду, наявності багатої екосистеми бібліотек для машинного навчання та обробки зображень, швидкості розробки та підтримки від активної спільноти розробників.

Ці переваги сприяють швидкому розвитку та ефективній імплементації алгоритмів розпізнавання облич у програмному забезпеченні.

3.2 Обґрунтування вибору середовища розробки

Інтегроване середовище розробки (IDE) - потужний інструмент, який полегшує життя програмістам під час написання коду. Такі середовища оснащені текстовими редакторами з багатьма корисними функціями, такими як налаштування плагінів, встановлення необхідних бібліотек та гарячих клавіш. Ці можливості IDE значно прискорюють процес створення програмних продуктів.

Під час вибору IDE для розробки WEB-ресурсів, розглядалися такі популярні редактори коду: Visual Studio Code, Sublime Text та PyCharm.

Visual Studio Code [11] - високопродуктивний редактор коду, розроблений легендарною компанією Microsoft, заснованою Біллом Гейтсом та Полом Аленом у 1975 році. Microsoft відома своїми операційними системами Windows, офісними пакетами Office, WEB-браузером Microsoft Edge та власним магазином додатків Microsoft Store. Однією з ключових цілей компанії є забезпечення сумісності між своїми продуктами та послугами, що надає користувачам більше зручності та можливостей.

Visual Studio Code є безкоштовним та відкритим для розробників, пропонуючи широкий спектр можливостей для програмування на різноманітних мовах (рис. 3.1).

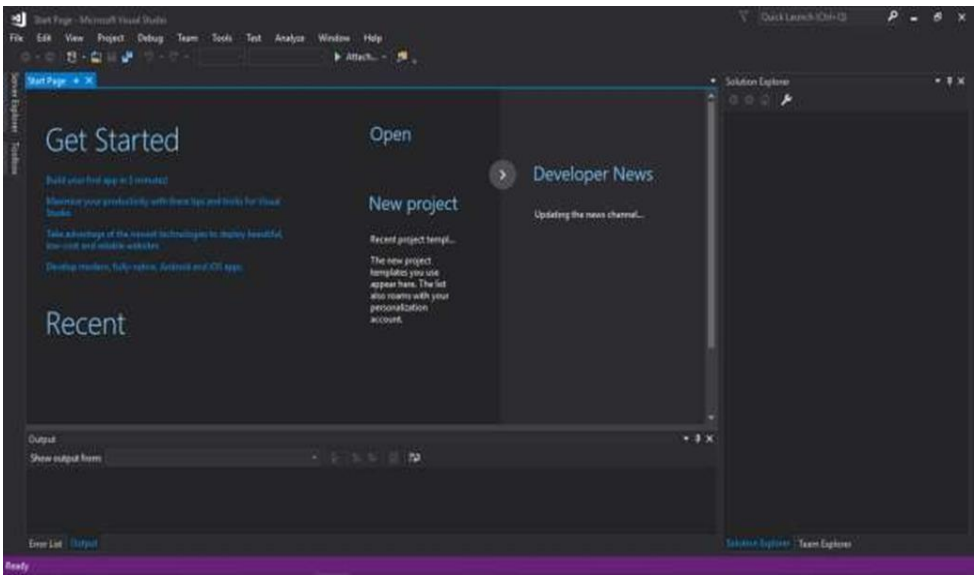

Рисунок 3.1 – Головне вікно редактора коду Visual Studio Code

Розглянемо ключові особливості редактора коду Visual Studio Code:

- Інтеграція: забезпечує сумісність з іншими сервісами та інструментами Microsoft, включаючи Git, Azure, TypeScript.
- Спільнота: має доброзичливу та активну спільноту, яка допомагає як досвідченим користувачам, так і новачкам вирішувати різні питання, пов'язані з кодом.
- Підтримка мов: підтримує безліч сучасних мов програмування, таких як JavaScript, Python, Java, C#, PHP та інші.
- Розширення: пропонує великий вибір розширень, які дозволяють налаштувати середовище відповідно до індивідуальних потреб.

Розглянемо недоліки:

- Нестабільність: іноді користувачі можуть стикатися з непередбачуваними збоями або нестабільністю при використанні певних розширень.

PyCharm [12] – потужне інтегроване середовище розробки (IDE) для мови програмування Python, розроблене компанією JetBrains. Воно пропонує багато функцій та інструментів, які допомагають підвищити продуктивність розробників Python.

Переваги PyCharm:

- Інтелектуальне автодоповнення коду та перевірка помилок: PyCharm забезпечує інтелектуальне автодоповнення коду, перевірку помилок та ефективну навігацію по коду, це економить час та підвищує якість коду.
- Підтримка різних фреймворків та бібліотек: PyCharm має вбудовану підтримку для популярних фреймворків та бібліотек Python, таких як Django, Flask, NumPy та багатьох інших.
- Інтеграція з системами контролю версій: PyCharm тісно інтегрується з системами контролю версій, такими як Git, Mercurial та Subversion, що полегшує роботу в команді.

Недоліки PyCharm:

- Висока вартість: PyCharm має платну професійну версію, яка може бути дорогою для деяких розробників або невеликих команд.

• Ресурсоємність: PyCharm може споживати значні обсяги оперативної пам'яті та процесорних ресурсів, особливо при роботі з великими проектами.

• Крива навчання: як і будь-яке потужне IDE, PyCharm може мати крутий крок навчання для початківців, які звикли до більш простих текстових редакторів.

На рисунку 3.2 подано загальний вигляд головного вікна редактора коду PyCharm.

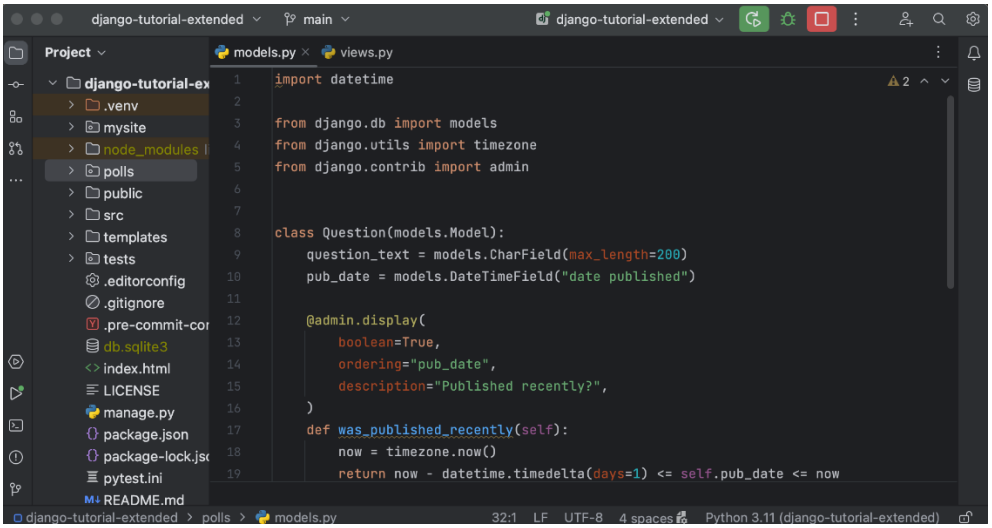

Рисунок 3.2 – Загальний вигляд головного вікна редактора коду PyCharm

Sublime Text [13] – текстовий редактор, сумісний з різними платформами, створений Джоном Скінером для розробки програмного забезпечення (рис. 3.3). Цей редактор широко використовується для роботи з різноманітними мовами програмування, включаючи Python, PHP, Ruby тощо.

Розглянемо основні переваги Sublime Text:

- Розширюваність: підтримує безліч плагінів, що дозволяють розширити функціональність редактора відповідно до потреб розробників;
- Налаштування: пропонує різноманітні можливості для налаштування, що забезпечує комфортне користування;
- Продуктивність: оптимізований для ефективної роботи на менш потужних комп’ютерах, що робить його ідеальним для великих проектів та обробки великих файлів.

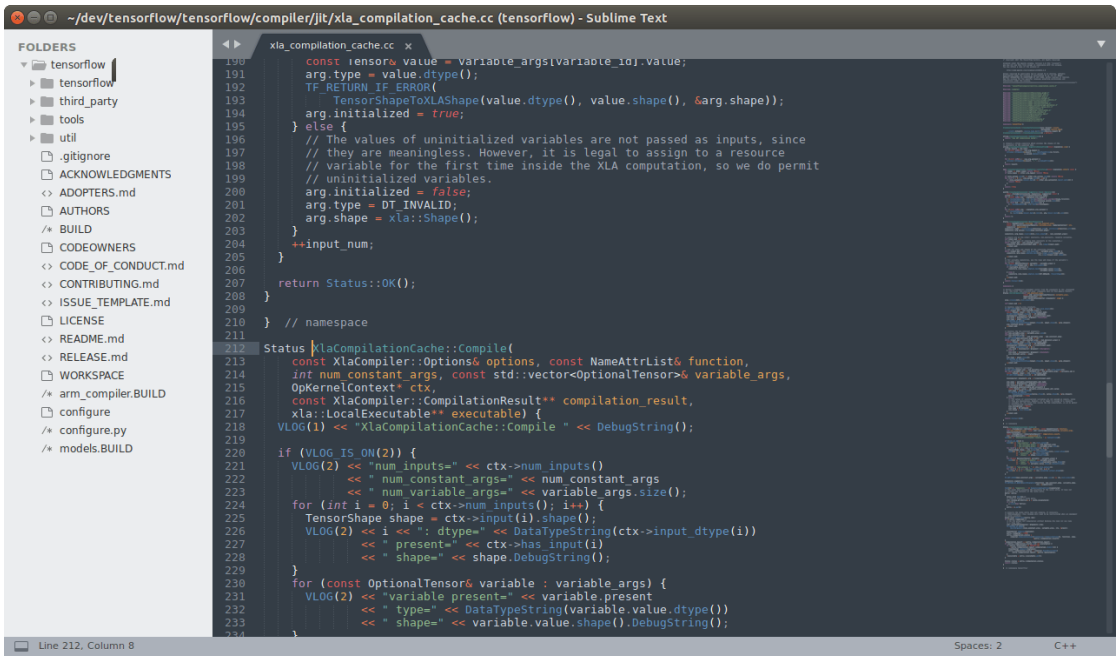

Рисунок 3.3 –Загальний вигляд головного вікна редактора коду Sublime Text

Недоліки Sublime Text:

- Платність: Sublime Text не є безкоштовним, однак доступна пробна версія, яка дозволяє користувачам ознайомитися з можливостями редактора перед покупкою;
- Оновлення: версія Sublime Text 3 тривалий час не отримувала оновлень, що могло стати проблемою для користувачів, які потребують регулярних оновлень і виправлення помилок.

В таблиці 3.2 подано порівняльні характеристики середовищ розробки програмного модуля для розпізнавання облич.

Система, над якою працюємо, є програмою для якої обрано мову програмування Python та відповідну бібліотеку OpenCV.

OpenCV – це бібліотека алгоритмів для обробки зображень та загальних алгоритмів з відкритим кодом, яку можна використовувати у наукових та комерційних цілях. Це міжплатформена бібліотека, яка дозволяє розробляти програми комп'ютерного зору в реальному часі. Основний акцент робиться на обробці зображень, захопленні та аналізі відео, включаючи функції, такі як виявлення облич та об'єктів [14].

Таблиця 3.2 – Порівняльні характеристики середовищ розробки

Характеристика	PyCharm	Visual Studio Code	Sublime Text
Інтеграція	Сумісність з інструментами JetBrains, Git, Docker, SQL	Інтеграція з Git, Azure, TypeScript	Підтримка плагінів для інтеграції
Спільнота	Велика активна спільнота	Активна і доброзичлива спільнота	Менш активна спільнота
Підтримка мов	Оптимізований для Python, підтримка інших мов	Підтримка численних мов програмування	Підтримка різних мов програмування
Розширення	Великий вибір плагінів	Широкий вибір розширень	Багато плагінів для розширення функцій
Налаштування	Багато можливостей налаштування	Багато налаштувань для користувачів	Пропонує різні налаштування
Продуктивність	Може бути важким для слабких ПК	Добре оптимізований	Оптимізований для слабких ПК
Платність	Community Edition безкоштовна, повна версія платна	Безкоштовний	Пробна версія, повна версія платна

Комп'ютерне бачення можна описати як дисципліну, яка досліджує, як перетворити та розуміти тривимірний простір за допомогою його двовимірних зображень. Це включає моделювання та реплікацію людського зору за допомогою комп'ютерного програмного та апаратного забезпечення.

Комп'ютерне бачення - це процес створення конкретних та значущих описів фізичних об'єктів на основі їх зображень. Результатом цього процесу є опис або інтерпретація структур у тривимірному просторі [15].

3.3 Опис програмних рішень

Першим кроком у процесі розпізнавання обличчя є виявлення їх на фотографії. Це може бути здійснено за допомогою камери смартфону або будь-якого іншого пристрою, який може автоматично виявляти обличчя перед зйомкою. Функція розпізнавання обличчя, хоч і призначена для камер, може бути використана для визначення ділянки зображення, яка потім буде використана для подальшого розпізнавання [16] (рис. 3.4).

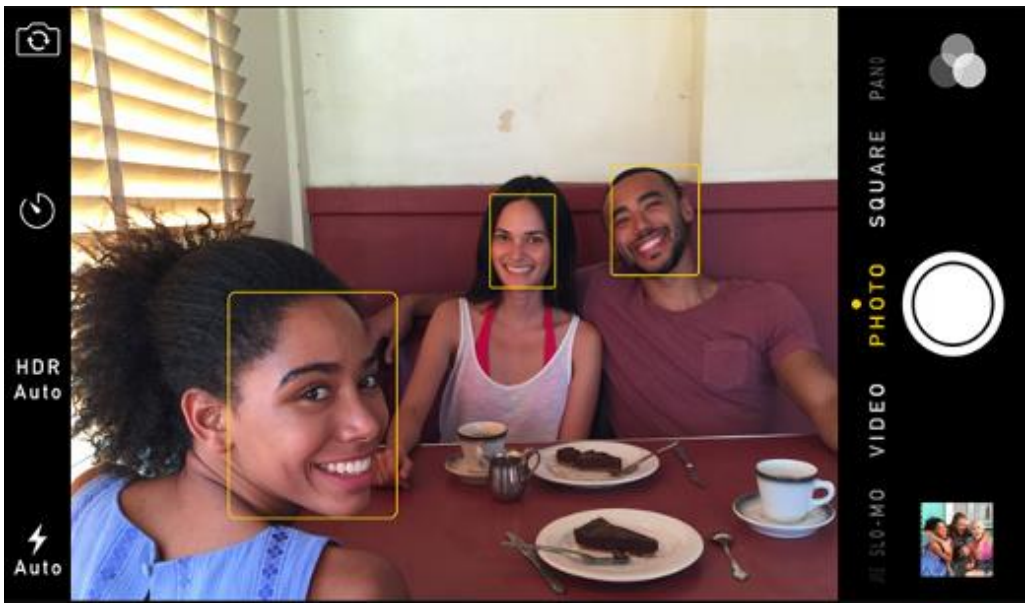

Рисунок 3.4 – Виділення облич

Технологія розпізнавання облич стала популярною на початку 2000-х років, коли Пол Віола та Майкл Джонс винайшли швидкий спосіб виявлення облич. Однак більш надійний метод, відомий як "Гістограма орієнтованих градієнтів" або HOG, був розроблений у 2005 році [17]. Для початку аналізу зображення перетворюється у чорно-біле, оскільки кольорові дані не потрібні для процесу розпізнавання обличчя. Потім кожен піксель зображення аналізується окремо для визначення напрямку зміни яскравості. Цей процес дозволяє створити градієнти для кожного пікселя, які показують напрямок зміни яскравості (рис. 3.5).

Рисунок 3.5 – Переведення у чорно-білий

Цей процес виконується за допомогою функції переведення зображення у чорно-білий колір, яка наявна в бібліотеці OpenCV (рис. 3.6).

```
def findEncodings(images):
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Рисунок 3.6 – Переведення у чорно-білий колір

Потім розглядається кожен піксель на зображенні по одному. У такому випадку для кожного окремого пікселя важливо розглянути пікселі, які безпосередньо його оточують (рис. 3.7).

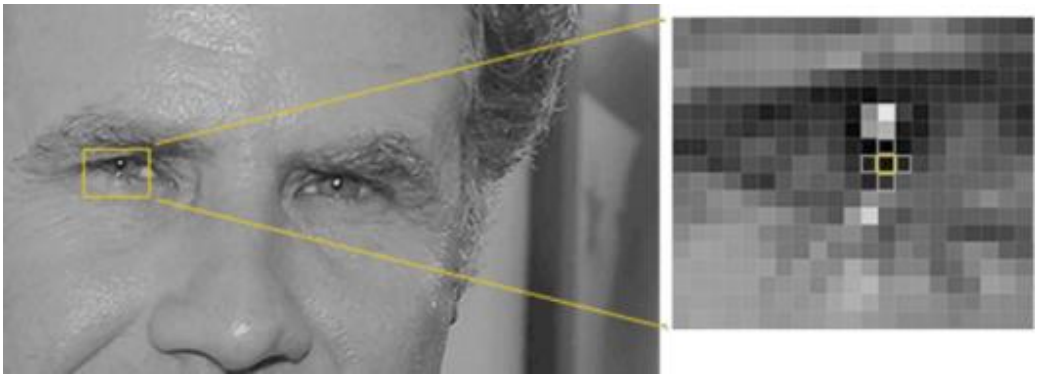

Рисунок 3.7 – Аналіз пікселів

Мета цього процесу – з’ясувати, наскільки темним є поточний піксель порівняно з пікселями, які безпосередньо його оточують. Потім розміщується стрілка, показуючи, в якому напрямку зображення стає темнішим (рис. 3.8).

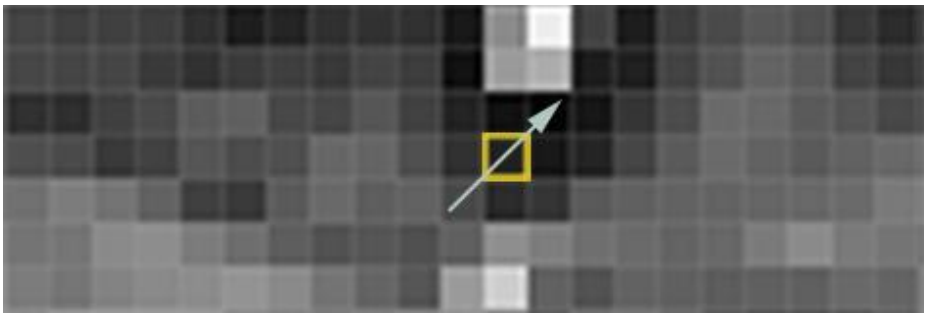

Рисунок 3.8 – Градієнти пікселів

Повторюють цей процес для кожного пікселя на зображенні, у результаті кожен піксель буде замінено стрілкою [18]. Ці стрілки називаються градієнтами, і вони показують потік від світлого до темного пікселя по всьому зображенню (рис. 3.8 − 3.10).

Рисунок 3.9 – Градієнти пікселів зображення

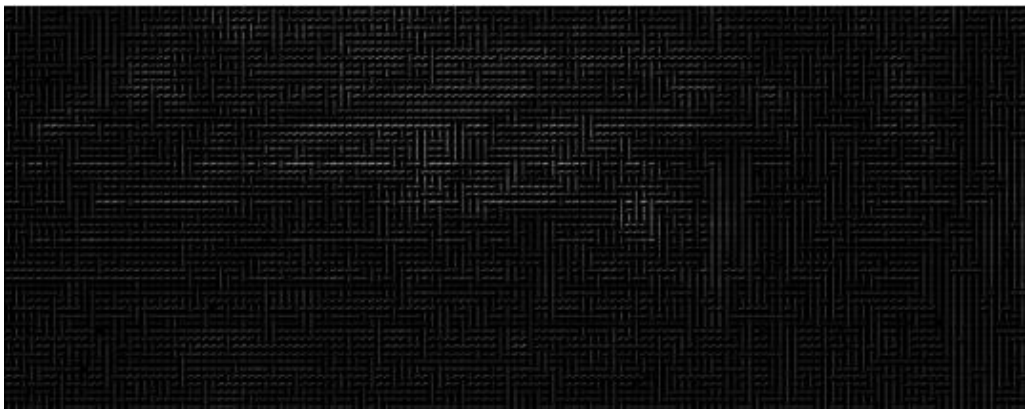

Рисунок 3.10 – Градієнти пікселів зображення

Існує важлива причина для того, щоб замінити значення пікселів на градієнти. Якщо аналізувати пікселі безпосередньо, дуже темні та дуже світлі зображення однієї і тієї ж особи матимуть абсолютно різні значення. Проте, з урахуванням напрямку зміни яскравості, як темні, так і світлі зображення отримають однакові результати.

Але збереження градієнтів для кожного пікселя може призвести до надмірної деталізації, що ускладнює подальше розпізнавання. Тому зображення розбивають на малі квадрати розміром 16×16 пікселів кожен. У кожному квадраті обчислюється скільки градієнтів вказують на загальний напрямок, і потім цей квадрат на зображенні замінюється напрямками стрілок, які переважають [19].

У результаті отримується просте зображення, яке фіксує основну структуру обличчя (рис. 3.11).

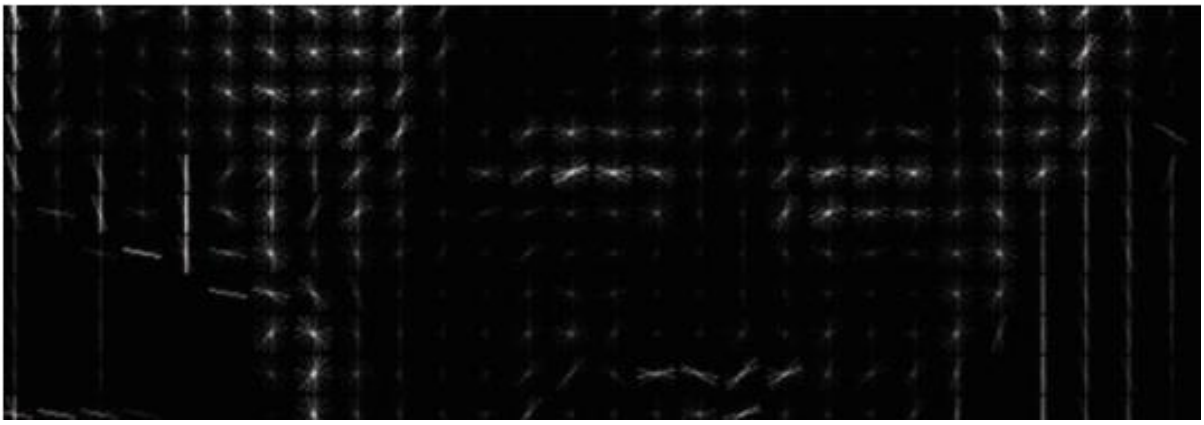

Рисунок 3.11 – Структура обличчя

Для того, щоб виявити обличчя на цьому зображенні за допомогою методу HOG, потрібно просто знайти частину іншого зображення, яка найбільш схожа на шаблон HOG, який вже відомий наразі і витягнути його з набору інших навчальних облич (рис. 3.12).

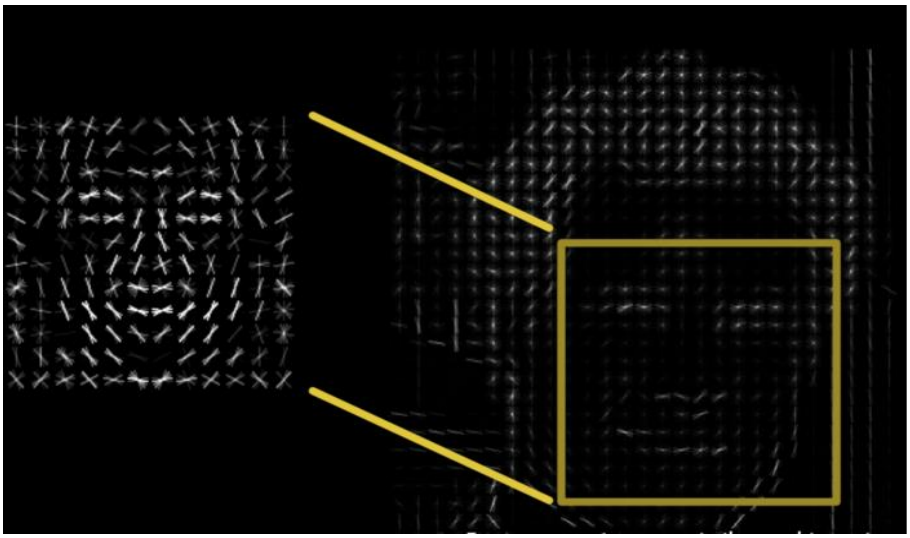

Рисунок 3.12 – Порівняння ознак

Для виконання попередніх завдань використовується функція виявлення облич у бібліотеці розпізнавання облич face_recognition, яка застосовує описаний метод детектування (рис. 3.13).

```
faceLoc = face_recognition.face_locations(imgRus)[0]
encodeElon = face_recognition.face_encodings(imgRus)[0]
cv2.rectangle(imgRus, (faceLoc[3], faceLoc[0]),
              (faceLoc[1], faceLoc[2]), (255, 0, 255), 2)

faceLocTest = face_recognition.face_locations(imgTest)[0]
encodeTest = face_recognition.face_encodings(imgTest)[0]
cv2.rectangle(imgTest,(faceLocTest[3],faceLocTest[0]),
              (faceLocTest[1],faceLocTest[2]),(255,0,255),2)
```

Рисунок 3.13 – Знаходження обличчя

Технологія дозволяє легко знаходити обличчя на будь-якому зображенні Потім можна приступити до написання основного кода у середовищі розробки Visual Studio Code [20].

Потрібно підготувати класифікатор, який може проводити вимірювання за новим тестовим зображенням і повідомляти, яка відома особа найбільш відповідає. Запуск цього класифікатора займає мілісекунди. Результатом класифікатора є ім’я людини.

Далі використовуються усі зазначені перед цим функції:

success, img = cap.read()

Читання кадру з WEB-камери. cap.read() повертає змінну success (індикатор успіху) та сам зображення img.

imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)

Зменшуємо зображення в 4 рази для пришвидшення обробки.

imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

Перетворення кольорової схеми з BGR (OpenCV за замовчуванням) на RGB (необхідно для face_recognition).

facesCurFrame = face_recognition.face_locations(imgS)

encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)

Використання face_recognition для знаходження місцезнаходжень облич та їхніх кодувань.

for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):

Проходимо через кожне знайдене обличчя.

matches = face_recognition.compare_faces(encodeListKnown, encodeFace) faceDis = face_recognition.face_distance(encodeListKnown, encodeFace) matchIndex = np.argmin(faceDis)

Порівняння кодування обличчя з відомими кодуваннями та обчислення відстані між ними. Знаходження індексу найменшої відстані.

if faceDis[matchIndex] < 0.50: name = classNames[matchIndex].upper() markAttendance(name) else: name = 'Unknown'

Якщо відстань менше 0.50, то обличчя вважається знайденим і його ім'я зберігається, інакше обличчя вважається невідомим.

y1, x2, y2, x1 = faceLoc y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4

Масштабування координат обличчя назад до оригінального розміру зображення.

cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)

cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED) cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)

Малювання прямокутника навколо обличчя та підписування імені (або "Unknown").

cv2.imshow('Webcam', img) cv2.waitKey(1)

Відображення зображення віконцем OpenCV. cv2.waitKey(1) чекає 1 мілісекунду, дозволяючи оновлювати зображення в реальному часі.

3.4 Тестування роботи програми та аналіз результатів

Тестування програми є ключовим етапом у процесі розробки програмного забезпечення. Воно спрямоване на перевірку відповідності програми вимогам та забезпечення її бездоганного та надійного функціонування. Основна мета тестування полягає у виявленні помилок, дефектів та проблем, які можуть виникати під час роботи програми.

Тестування дозволяє перевірити правильність роботи програми у різних сценаріях та умовах експлуатації. Це допомагає виявити та виправити помилки до впровадження програми, що забезпечує її якість та надійність. Виявлені та виправлені помилки підвищують ефективність програми та задоволення користувачів від її використання [21].

Крім того, тестування допомагає гарантувати, що програма відповідає вимогам замовника та стандартам якості. Було проведено детальне тестування розробленого програмного модуля для розпізнавання облич (понад 100 тестових запусків) (рис.3.14).

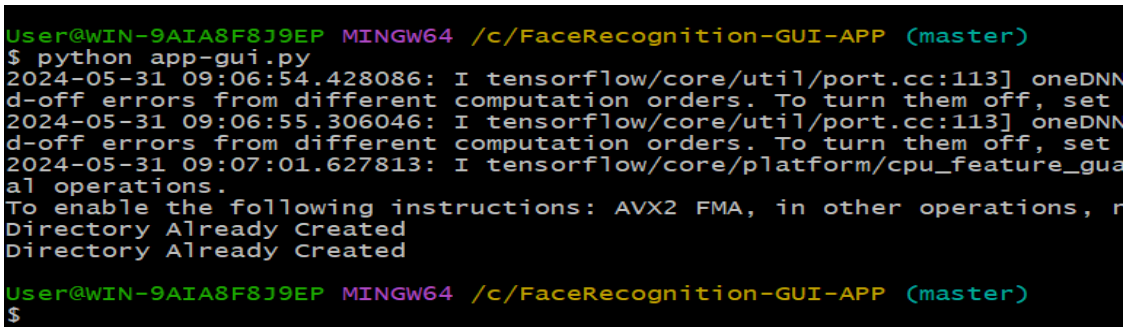

Рисунок 3.14 – Запуск програмного модуля розпізнавання облич

В робочому вікні знаходяться кнопки «Sign Up», яка сприяє введення імені користувача, сканування обличчя та його подальшої обробки для ідентифікації осіб, а також кнопка «Check a User» (рис. 3.15), яка відкриває список користувачів. Програмний функціонал дозволяє додавати або видаляти користувачів з бази даних, а також завантажувати їх готові профілі (рис. 3.16).

Рисунок 3.15 – Загальний вид програмного модуля для розпізнавання облич

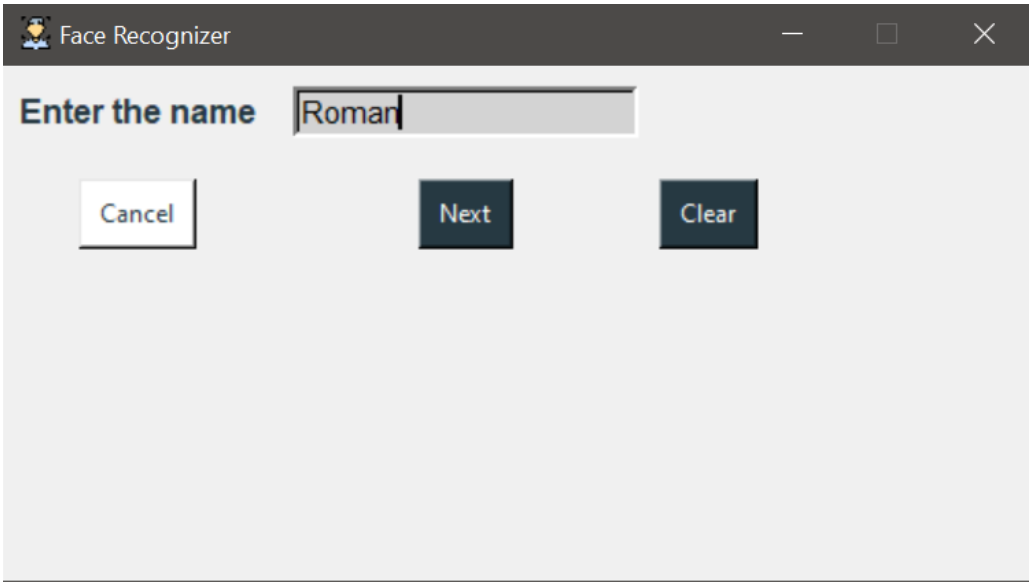

Рисунок 3.16 – Вікно введення даних користувача

Кнопка «Capture Data Set» запускає програмний модуль та камеру для розпізнавання облич користувачів, нейронній мережі потрібно 301 фотографія для детального обчислення вхідних даних та формування профілю авторизованого користувача (рис. 3.17).

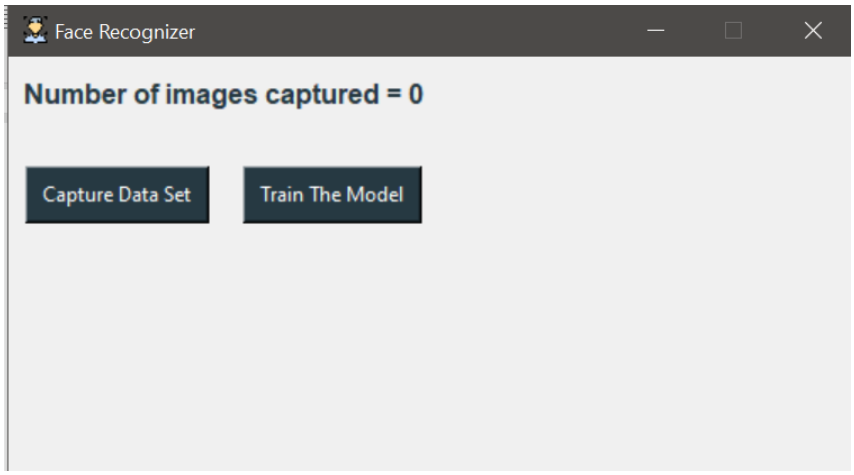

Рисунок 3.17 – Вікно функції тренування моделі

За допомогою кнопки «Train the model» користувач інформується про успішне розпізнавання обличя (рис. 3.18).

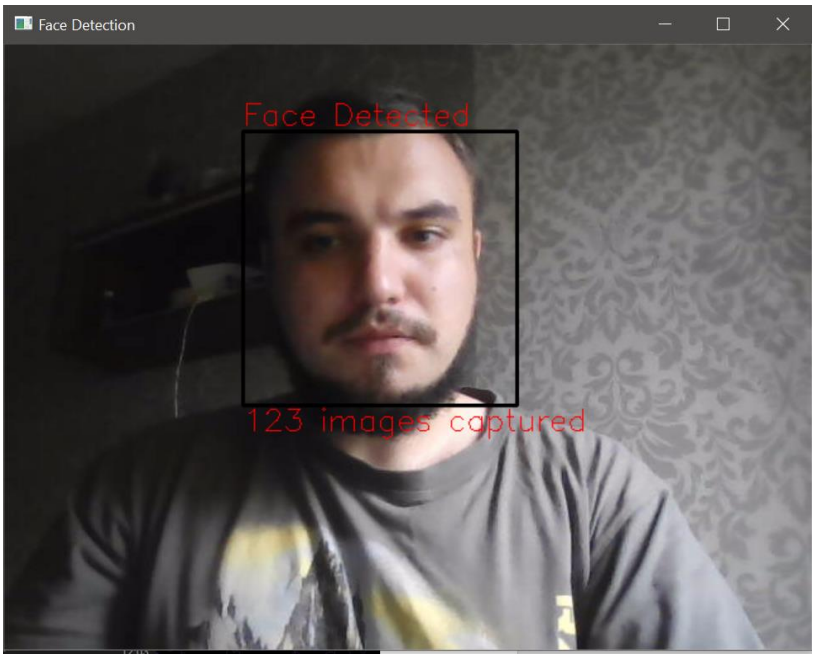

Рисунок 3.18 – Процес тренуваня моделі

Після успішного створення профілю та генерації коду за допомогою програмного модуля розпізнавання облич, користувач верифікує свою особу та отримує готовий унікальний ключ в реальному часі (рис. 3.19, 3.20).

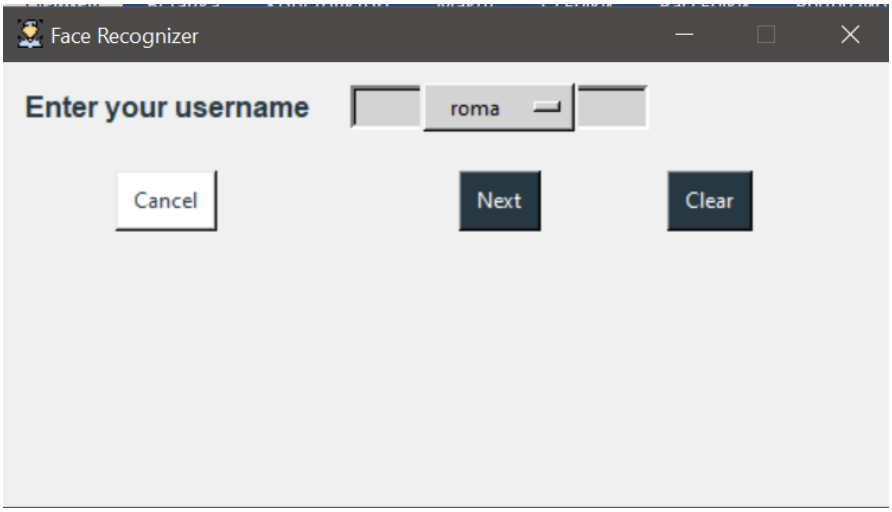

Рисунок 3.19 – Вибір користувача

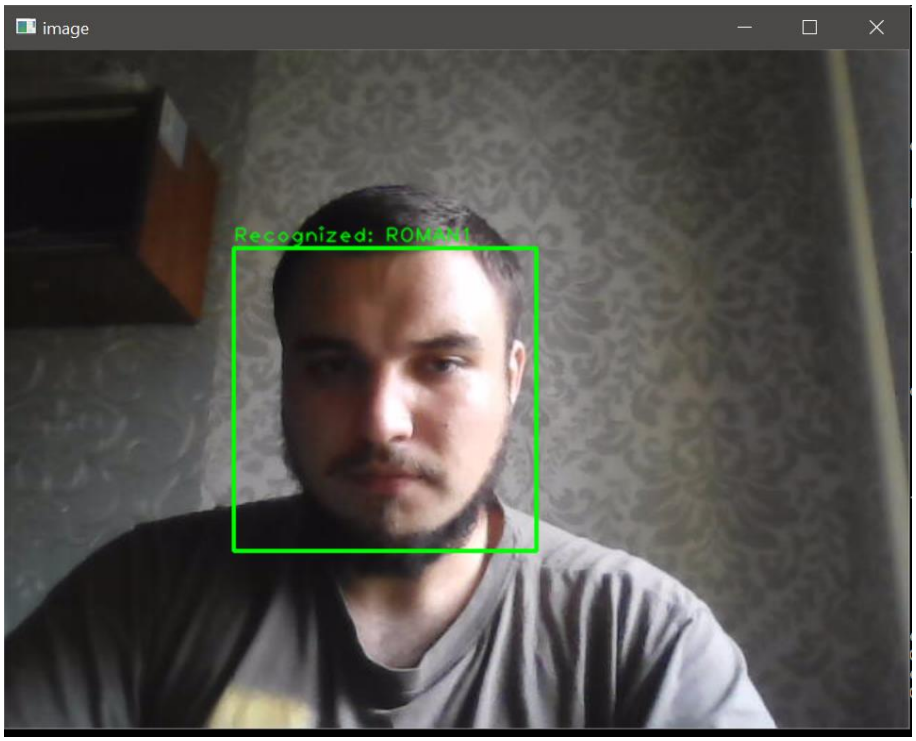

Рисунок 3.20 – Ініціалізація обличя по тренованій моделі

Розроблений програмний модуль розпізнавання облич успішно пройшов тестування та відповідає всім заданим вимогам.

У таблиці 3.3 наведено результати тестування розробленого програмного продукту та його аналогів.

В результаті розробки було реалізовано функціонал, який був відсутній в системах-аналогах, а саме:

- розпізнавання більше 1-го обличя;
- розпізнавання облич на відео за посиланням.

Таблиця 3.3 – Результати тестування розробленого програмного модуля та аналогів

	Наянвність функції «розпізнавання більше 1-го обличя»	Ідентифікація осіб	Розпізнавання групи осіб	Інтеграція з іншими системами	Реєстрація нових осіб
Розроблюваний модуль	+	+	+	+	+
OpenCV	+	-	+	+	-
Google Cloud Vision API	-	+	-	-	+
Microsoft Azure Face API	-	+	+	_	_

Отже, проаналізувавши та протестувавши розроблений програмний продукт, можна дійти висновку, що поставлених цілей було досягнуто.

З таблиці 3.3 можна побачити, що розроблений програмний модуль розпізнавання облич має покращений функціонал у порівнянні з системами-аналогами щонайменше на 2 можливості. Розширений функціонал сприятиме в задоволенні користувачів від використання цього програмного продукту.

3.5 Висновок

В результаті проведеного аналізу та тестування розробленого програмного модуля розпізнавання облич можна стверджувати, що поставлені цілі були досягнуті. Модуль є безкоштовним для користувачів і легко інтегрується, це робить його доступним і зручним для використання. Програмний модуль також є кросплатформеним, тобто може бути використаний на різних пристроях та операційних системах.

Результати порівняння з аналогічними модулями свідчать про значне покращення функціоналу розробленого програмного продукту. Він надає користувачам щонайменше 2 додаткові можливості, що суттєво підвищує їх продуктивність і задоволення від використання модуля. Крім того, розроблене рішення забезпечує комфортне та універсальне середовище для користувачів.

Модуль має більший функціонал, включаючи додаткову функцію «розпізнавання облич по посиланням» та «розпізнавання більше 1 обличя», що дозволяє швидко розпізнавати обличчя, які є на відео не залежно чи це прямий потік чи будь-яке відео. Це розширює можливості користувачів і надає їм більше корисної інформації.

Таким чином, результати аналізу та тестування підтверджують успішну реалізацію поставлених цілей, що робить розроблений програмний модуль привабливим і вигідним для користувачів.

ВИСНОВКИ

Всі задачі, поставлені для реалізації програмного модуля для розпізнавання облич були виконані в повному обсязі, а саме: обґрунтувана доцільність розробки програмного модуля для розпізнавання облич; спроектувано програмний модуль для розпізнавання облич; програмно реалізувано модуль для розпізнавання облич; виконано тестування програми та проведено аналіз отриманих результатів; розроблено інструкцію користувача.

Обґрунтування доцільності розробки такого модуля базувалося на його потенційній корисності у різних сферах, від безпеки до розваг, а також на актуальності та популярності подібних технологій у сучасному світі.

Під час проектування модуля було використано методи, які дозволили створити чітку структуру та алгоритм його роботи. У результаті була розроблена загальна UML-діаграма класів, UML-діаграма послідовності та схема алгоритму, що забезпечило зручність та ефективність подальшого програмного втілення.

Для реалізації були обрані мови програмування Python з використанням бібліотеки OpenCV. Використання цих інструментів дозволило забезпечити високу швидкість та точність розпізнавання облич, а також зручний інтерфейс для користувача.

Під час порівняння з аналогічними програмами було виявлено, що розроблений програмний модуль для розпізнавання облич має розширений функціонал, який перевершує програмні аналоги, принаймні на 2 можливості.

Мета роботи була досягнута шляхом розширення функціональних можливостей програмного модуля розпізнавання облич. У порівнянні з аналогами, розроблений модуль включає додаткову функцію «розпізнавання більше 1 обличя» та «розпізнавання облич на відео», що забезпечує більшу інформативність.

Отже, ціль роботи, яка полягала в розширенні функціональних можливостей програмного модуля для розпізнавання облич, була повністю досягнута. Розроблений модуль виконує всі заплановані завдання і виходить за межі функціоналу аналогічних програм.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Собщенко Р. О., Крилик Л. В. Перспективи розробки програмного модуля розпізнавання облич. *LII Міжнародна науково-практична інтернет-конференція студентів аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2024), м. Вінниці у 2024 р.* URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21059 (дата звернення: 31.05.2024).

2. Spaun N. 2nd ieee international conference on biometrics theory applications and systems. URL: https://cse.nd.edu/BTAS_08/ (дата звернення: 31.05.2024).

3. Fairhurst M. Seventh International Conference of Document Analysis and Recognition. URL: https://ieeexplore.ieee.org/document/1227617/keywords#keywords (дата звернення: 03.06.2024).

4. Schoutien B. Biometrics and their use in e-passports Schoutien. URL: https://research.tue.nl/en/publications/biometrics-and-their-use-in-e-passports (дата звернення: 03.06.2024).

5. ISO/IEC 19785-1. URL: https://www.iso.org/standard/66179.html (дата звернення: 05.06.2024).

6. Огляд біометричних методів ідентифікації особистості. URL: http://masters.donntu.org/2013/fknt/fomenko/library/article3.htm (дата звернення: 08.06.2024).

7. Biometric cryptosystems issues and challenges. URL: https://ieeexplore.ieee.org/document/1299169 (дата звернення: 09.06.2024).

8. Fuzzy Extractors for Minutiae-Based Fingerprint Authentication. URL: https://link.springer.com/chapter/10.1007/978-3-540-74549-5_80 (дата звернення: 09.06.2024).

9. Binary feature vector fingerprint representation from minutiae vicinities. URL: https://ieeexplore.ieee.org/document/563448880 (дата звернення: 10.06.2024).

10. Нейромережевий захист персональних біометричних даних. URL: https://azon.market/knigi/kompyyuternaya-literatura/neyrosetevaya zaschitapersonalnyih-biometricheskih-dannyih---volchihin-vi mysh2567676?limit=100 (дата звернення: 10.06.2024).

11. Tcутома M. A. Case Study for User Identificati. URL: http://web.mit.edu/6.857/OldStuff/Fall03/ref/gummy-slides.pdf. (дата звернення: 10.06.2024).

12. Reuters. URL: https://www.reuters.com/video/watch/peace-signs-risk-fingerprint-theft-saysid370920514 (дата звернення: 10.06.2024).

13. Chaos Communication Congress. URL: https://www.theguardian.com/technology/2014/dec/30/hackerfakesgerman-ministers-fingerprints-using-photos-of-her-hands (дата звернення: 12.06.2024).

14. Roy A. MasterPrint: Exploring the Vulnerability of Partial FingerprintBased Authentication Systems. URL: https://ieeexplore.ieee.org/document/7893784/authors#authors (дата звернення: 12.06.2024).

15. Parsons A. Paying for a pint with your finger: The tech that could kill off Cards. URL: https://news.sky.com/story/paying-for-a-pint-with-your-finger-the-tech-that-couldkill-off-cards-10780629 (дата звернення: 12.06.2024).

16. Brewster T. We Broke Into A Bunch Of Android Phones With A 3D-Printed. URL: https://www.forbes.com/sites/thomasbrewster/2018/12/13/we-broke-into-a-bunch-ofandroid-phones-with-a-3d-printed-head/?sh=2a17646c1330 (дата звернення: 13.06.2024).

17. Hopfield J. Neural networks and physical systems with emergent collective computational abilities. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC346238 (дата звернення: 13.06.2024).

18. Wiley J. The Organization of Behavior. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cne.900930310 (дата звернення: 13.06.2024).

19. Застосування штучного інтелекту. URL: https://uk.m.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D1%88%D1%82%D1%83%D1%87%D0%BD%D0%BE%D0%B3%D0%BE_%D1%96%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D1%83 (дата звернення: 13.06.2024).

20. Dodis Y. Fuzzy extractors how to generate strong keys from biometrics and other noisy data. URL: https://link.springer.com/chapter/10.1007/978-3-540-24676-3_31 (дата звернення: 13.06.2024).

21. ISO/IEC 19794-1. URL: https://www.iso.org/standard/50862.html (дата звернення: 13.06.2024).

ДОДАТКИ

ДОДАТОК А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Розробка програмного модуля для розпізнавання облич

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп`ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 89,1% Схожість 10,9%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Собщенко Р.О.

Керівник роботи _____ Крилик Л.В.

ДОДАТОК Б (обов’язковий)

Лістинг програми

```
from Detector import main_app
from create_classifier import train_classifer
from create_dataset import start_capture
import tkinter as tk
from tkinter import font as tkfont
from tkinter import messagebox,PhotoImage
#from PIL import ImageTk, Image
from gender_prediction import emotion,ageAndgender
names = set()


class MainUI(tk.Tk):

    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)
        global names
        with open("nameslist.txt", "r") as f:
            x = f.read()
            z = x.rstrip().split(" ")
            for i in z:
                names.add(i)
        self.title_font = tkfont.Font(family='Helvetica', size=16, weight="bold")
        self.title("Face Recognizer")
        self.resizable(False, False)
        self.geometry("500x250")
        self.protocol("WM_DELETE_WINDOW", self.on_closing)
```

```
        self.active_name = None
        container = tk.Frame(self)
        container.grid(sticky="nsew")
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)
        self.frames = {}
        for F in (StartPage, PageOne, PageTwo, PageThree, PageFour):
            page_name = F.__name__
            frame = F(parent=container, controller=self)
            self.frames[page_name] = frame
            frame.grid(row=0, column=0, sticky="nsew")
        self.show_frame("StartPage")

    def show_frame(self, page_name):
            frame = self.frames[page_name]
            frame.tkraise()

    def on_closing(self):

        if messagebox.askokcancel("Quit", "Are you sure?"):
            global names
            f =  open("nameslist.txt", "a+")
            for i in names:
                    f.write(i+" ")
            self.destroy()


class StartPage(tk.Frame):

        def __init__(self, parent, controller):
```

```
            tk.Frame.__init__(self, parent)
            self.controller = controller
            #load = Image.open("homepagepic.png")
            #load = load.resize((250, 250), Image.ANTIALIAS)
            render = PhotoImage(file='homepagepic.png')
            img = tk.Label(self, image=render)
            img.image = render
            img.grid(row=0, column=1, rowspan=4, sticky="nsew")
            label = tk.Label(self, text="                Home Page                ",
font=self.controller.title_font,fg="#263942")
            label.grid(row=0, sticky="ew")
            button1 = tk.Button(self, text="        Sign up      ", fg="#ffffff",
bg="#263942",command=lambda: self.controller.show_frame("PageOne"))
            button2 = tk.Button(self, text="      Check a User    ", fg="#ffffff",
bg="#263942",command=lambda: self.controller.show_frame("PageTwo"))
            button3 = tk.Button(self, text="Quit", fg="#263942", bg="#ffffff",
command=self.on_closing)
            button1.grid(row=1, column=0, ipady=3, ipadx=7)
            button2.grid(row=2, column=0, ipady=3, ipadx=2)
            button3.grid(row=3, column=0, ipady=3, ipadx=32)


        def on_closing(self):
            if messagebox.askokcancel("Quit", "Are you sure?"):
                global names
                with open("nameslist.txt", "w") as f:
                    for i in names:
                        f.write(i + " ")
                self.controller.destroy()
```

```
class PageOne(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller
        tk.Label(self, text="Enter the name", fg="#263942", font='Helvetica 12 bold').grid(row=0, column=0, pady=10, padx=5)
        self.user_name = tk.Entry(self, borderwidth=3, bg="lightgrey", font='Helvetica 11')
        self.user_name.grid(row=0, column=1, pady=10, padx=10)
        self.buttoncanc = tk.Button(self, text="Cancel", bg="#ffffff", fg="#263942", command=lambda: controller.show_frame("StartPage"))
        self.buttonext = tk.Button(self, text="Next", fg="#ffffff", bg="#263942", command=self.start_training)
        self.buttonclear = tk.Button(self, text="Clear", command=self.clear, fg="#ffffff", bg="#263942")
        self.buttoncanc.grid(row=1, column=0, pady=10, ipadx=5, ipady=4)
        self.buttonext.grid(row=1, column=1, pady=10, ipadx=5, ipady=4)
        self.buttonclear.grid(row=1, ipadx=5, ipady=4, column=2, pady=10)
    def start_training(self):
        global names
        if self.user_name.get() == "None":
            messagebox.showerror("Error", "Name cannot be 'None'")
            return
        elif self.user_name.get() in names:
            messagebox.showerror("Error", "User already exists!")
            return
        elif len(self.user_name.get()) == 0:
            messagebox.showerror("Error", "Name cannot be empty!")
            return
```

```
            name = self.user_name.get()
            names.add(name)
            self.controller.active_name = name
            self.controller.frames["PageTwo"].refresh_names()
            self.controller.show_frame("PageThree")

        def clear(self):
            self.user_name.delete(0, 'end')


    class PageTwo(tk.Frame):

        def __init__(self, parent, controller):
            tk.Frame.__init__(self, parent)
            global names
            self.controller = controller
            tk.Label(self, text="Enter your username", fg="#263942", font='Helvetica 12
bold').grid(row=0, column=0, padx=10, pady=10)
            self.user_name = tk.Entry(self, borderwidth=3, bg="lightgrey",
font='Helvetica 11')
            self.user_name.grid(row=0, column=1, pady=10, padx=10)
            self.buttoncanc = tk.Button(self, text="Cancel", command=lambda:
controller.show_frame("StartPage"), bg="#ffffff", fg="#263942")
            self.buttonclear = tk.Button(self, text="Clear", command=self.clear,
fg="#ffffff", bg="#263942")
            self.menuvar = tk.StringVar(self)
            self.dropdown = tk.OptionMenu(self, self.menuvar, *names)
            self.dropdown.config(bg="lightgrey")
            self.dropdown["menu"].config(bg="lightgrey")
```

```
        self.buttonext = tk.Button(self, text="Next", command=self.next_foo, fg="#ffffff", bg="#263942")
        self.dropdown.grid(row=0, column=1, ipadx=8, padx=10, pady=10)
        self.buttoncanc.grid(row=1, ipadx=5, ipady=4, column=0, pady=10)
        self.buttonext.grid(row=1, ipadx=5, ipady=4, column=1, pady=10)
        self.buttonclear.grid(row=1, ipadx=5, ipady=4, column=2, pady=10)

    def next_foo(self):
        if self.user_name.get() == 'None':
            messagebox.showerror("ERROR", "Name cannot be 'None'")
            return
        self.controller.active_name = self.user_name.get()
        self.controller.show_frame("PageFour")

    def clear(self):
        self.user_name.delete(0, 'end')

    def nextfoo(self):
        if self.menuvar.get() == "None":
            messagebox.showerror("ERROR", "Name cannot be 'None'")
            return
        self.controller.active_name = self.menuvar.get()
        self.controller.show_frame("PageFour")

    def refresh_names(self):
        global names
        self.menuvar.set('')
        self.dropdown['menu'].delete(0, 'end')
        for name in names:
```

```
                self.dropdown['menu'].add_command(label=name,
command=tk._setit(self.menuvar, name))

        class PageThree(tk.Frame):

            def __init__(self, parent, controller):
                tk.Frame.__init__(self, parent)
                self.controller = controller
                self.numimglabel = tk.Label(self, text="Number of images captured = 0",
font='Helvetica 12 bold', fg="#263942")
                self.numimglabel.grid(row=0, column=0, columnspan=2, sticky="ew",
pady=10)
                self.capturebutton = tk.Button(self, text="Capture Data Set", fg="#ffffff",
bg="#263942", command=self.capimg)
                self.trainbutton = tk.Button(self, text="Train The Model", fg="#ffffff",
bg="#263942",command=self.trainmodel)
                self.capturebutton.grid(row=1, column=0, ipadx=5, ipady=4, padx=10,
pady=20)
                self.trainbutton.grid(row=1, column=1, ipadx=5, ipady=4, padx=10,
pady=20)

            def capimg(self):
                self.numimglabel.config(text=str("Captured Images = 0 "))
                messagebox.showinfo("INSTRUCTIONS", "We will Capture 300 pic of your
Face.")
                x = start_capture(self.controller.active_name)
                self.controller.num_of_images = x
                self.numimglabel.config(text=str("Number of images captured = "+str(x)))

            def trainmodel(self):
```

```
            if self.controller.num_of_images < 300:
                messagebox.showerror("ERROR", "Not enough Data, Capture at least 300 images!")
                return
            train_classifer(self.controller.active_name)
            messagebox.showinfo("SUCCESS", "The model has been successfully trained!")
            self.controller.show_frame("PageFour")


        class PageFour(tk.Frame):

            def __init__(self, parent, controller):
                tk.Frame.__init__(self, parent)
                self.controller = controller

                label = tk.Label(self, text="Face Recognition", font='Helvetica 16 bold')
                label.grid(row=0,column=0, sticky="ew")
                button1 = tk.Button(self, text="Face Recognition", command=self.openwebcam, fg="#ffffff", bg="#263942")
                #button2 = tk.Button(self, text="Emotion Detection", command=self.emot, fg="#ffffff", bg="#263942")
                #button3 = tk.Button(self, text="Gender and Age Prediction", command=self.gender_age_pred, fg="#ffffff", bg="#263942")
                button4 = tk.Button(self, text="Go to Home Page", command=lambda: self.controller.show_frame("StartPage"), bg="#ffffff", fg="#263942")
                button1.grid(row=1,column=0, sticky="ew", ipadx=5, ipady=4, padx=10, pady=10)
                #button2.grid(row=1,column=1, sticky="ew", ipadx=5, ipady=4, padx=10, pady=10)
```

```
            #button3.grid(row=2,column=0, sticky="ew", ipadx=5, ipady=4, padx=10, pady=10)
            button4.grid(row=1,column=1, sticky="ew", ipadx=5, ipady=4, padx=10, pady=10)

        def openwebcam(self):
            main_app(self.controller.active_name)

        '''
        def gender_age_pred(self):
          ageAndgender()
        def emot(self):
            emotion()
    '''


    app = MainUI()
    app.iconphoto(True, tk.PhotoImage(file='icon.ico'))
    app.mainloop()
```

createclasiffer.py

```
import numpy as np
from PIL import Image
import os, cv2



# Method to train custom classifier to recognize face
def train_classifer(name):
    # Read all the images in custom data-set
```

```
path = os.path.join(os.getcwd()+"/data/"+name+"/")

faces = []
ids = []
labels = []
pictures = {}

# Store images in a numpy format and ids of the user on the same index in imageNp and id lists

for root,dirs,files in os.walk(path):
    pictures = files


for pic in pictures :

    imgpath = path+pic
    img = Image.open(imgpath).convert('L')
    imageNp = np.array(img, 'uint8')
    id = int(pic.split(name)[0])
    #names[name].append(id)
    faces.append(imageNp)
    ids.append(id)

ids = np.array(ids)

#Train and save classifier
clf = cv2.face.LBPHFaceRecognizer_create()
clf.train(faces, ids)
clf.write("./data/classifiers/"+name+"_classifier.xml")
```

```
#train_classifer('tho1')

```

create_dataset.py

```
import cv2
import os

def start_capture(name):
    path = "./data/" + name
    num_of_images = 0
    detector = cv2.CascadeClassifier("./data/haarcascade_frontalface_default.xml")
    try:
        os.makedirs(path)
    except:
        print('Directory Already Created')
    vid = cv2.VideoCapture(0)
    while True:

        ret, img = vid.read()
        new_img = None
        grayimg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        face = detector.detectMultiScale(image=grayimg, scaleFactor=1.1, minNeighbors=5)
        for x, y, w, h in face:
            cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 0), 2)
            cv2.putText(img, "Face Detected", (x, y-5), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255))
            cv2.putText(img, str(str(num_of_images)+" images captured"), (x, y+h+20), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255))
```

```
                new_img = img[y:y+h, x:x+w]
            cv2.imshow("Face Detection", img)
            key = cv2.waitKey(1) & 0xFF
            try :
                cv2.imwrite(str(path+"/"+str(num_of_images)+name+".jpg"),
new_img)
                num_of_images += 1
            except :

                pass
            if key == ord("q") or key == 27 or num_of_images > 300: #take 300 frames
                break
        cv2.destroyAllWindows()
        return num_of_images
    #take frames by extract a video
    def take_video(name, video):
        path = "./data/" + name
        num_of_images = 0
        detector = cv2.CascadeClassifier("./data/haarcascade_frontalface_default.xml")
        try:
            os.makedirs(path)
        except:
            print('Directory Already Created')
        vid = cv2.VideoCapture(video)
        if not vid.isOpened():
            print("Error: Could not open video file.")
            exit()
        num_of_images = 0
        while True:
            ret, img = vid.read()
```

```
            new_img = None
            grayimg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            face = detector.detectMultiScale(image=grayimg, scaleFactor=1.1, minNeighbors=5)
            if not ret:
                break  # Break the loop if no more frames are available
            for x, y, w, h in face:

                cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 0), 2)
                cv2.putText(img, "Face Detected", (x, y-5), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255))
                cv2.putText(img, str(str(num_of_images)+" images captured"), (x, y+h+20), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255))
                new_img = img[y:y+h, x:x+w]
            cv2.imshow("Face Detection", img)
            key = cv2.waitKey(1) & 0xFF
            try :
                cv2.imwrite(str(path+"/"+str(num_of_images)+name+".jpg"), new_img)
                num_of_images += 1
            except :

                pass
            if key == ord("q") or key == 27 or num_of_images > 300: #take 300 frames
                break
        vid.release()
        cv2.destroyAllWindows()
        return num_of_images


    #take_video('tho1', 'data\WIN_20230920_07_56_11_Pro.mp4')
```

ДОДАТОК В (обов'язковий)

ГРАФІЧНА ЧАСТИНА

«РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ ДЛЯ РОЗПІЗНАВАННЯ ОБЛИЧ»

Виконав: студент 4 курсу, групи 3КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Собщенко Р. О.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Крилик Л. В.
(прізвище та ініціали)

« 07 » червня 2024 р.

Вінниця ВНТУ - 2024 рік

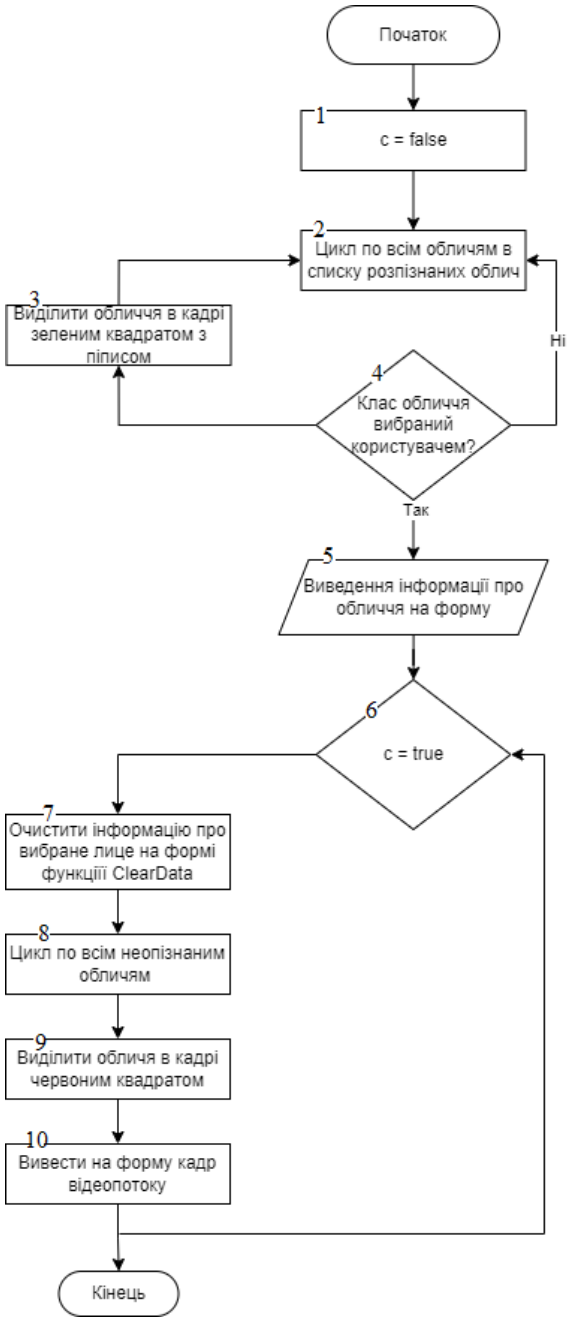

Рисунок В.1 – Схема алгоритму роботи програмного модуля для розпізнавання облич

Рисунок В.2 − Загальна структурна схема функціонування системи

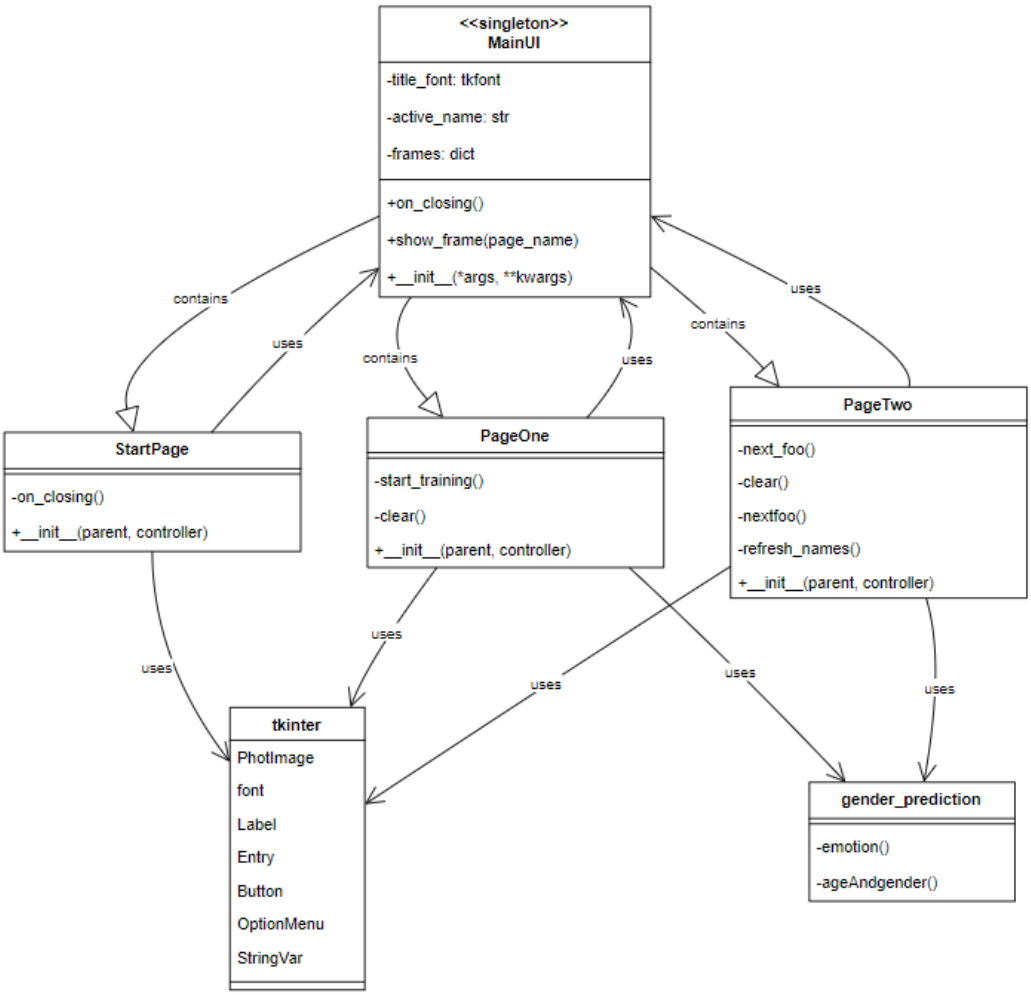

Рисунок В.3 – Загальна UML-діаграма класів

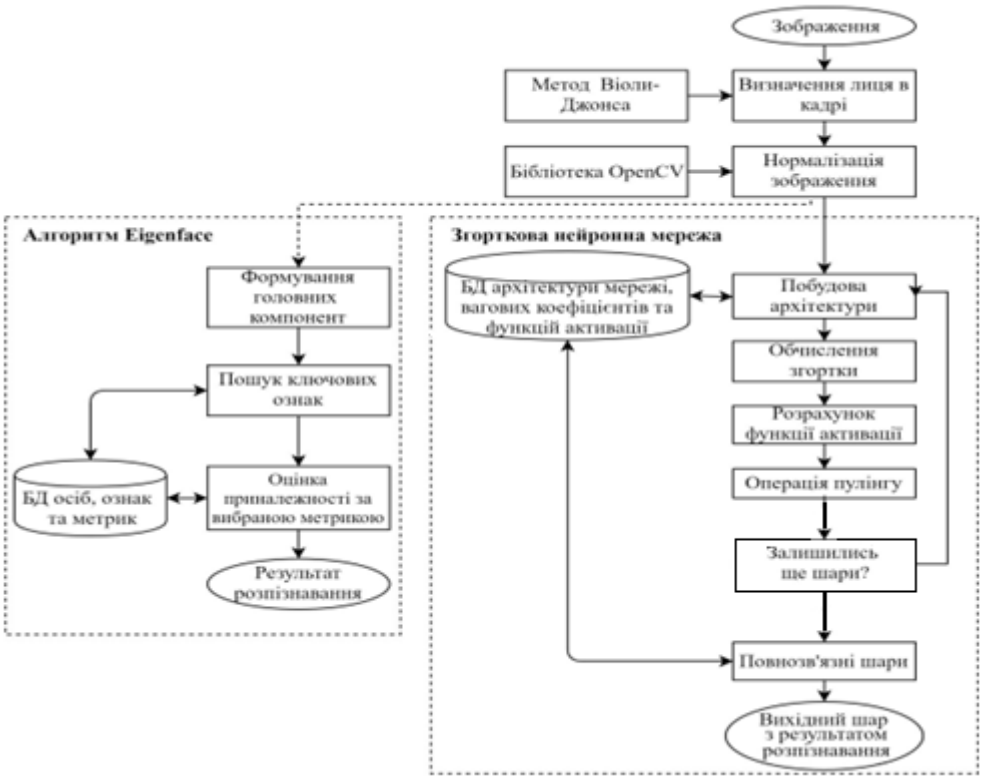

Рисунок В.4 − Структура принципу роботи програмного модуля для розпізнавання облич

Рисунок В.5 – UML-діаграма комунікації системи програмного модуля для розпізнавання облич

Рисунок В.6 – Загальний вид програмного модуля для розпізнавання облич

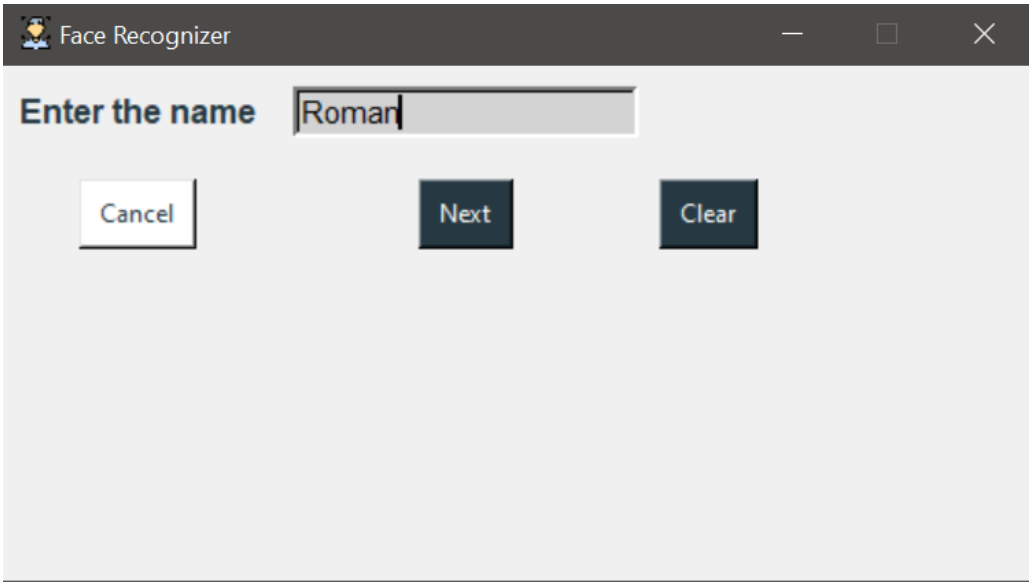

Рисунок В.7 – Вікно введення даних користувача

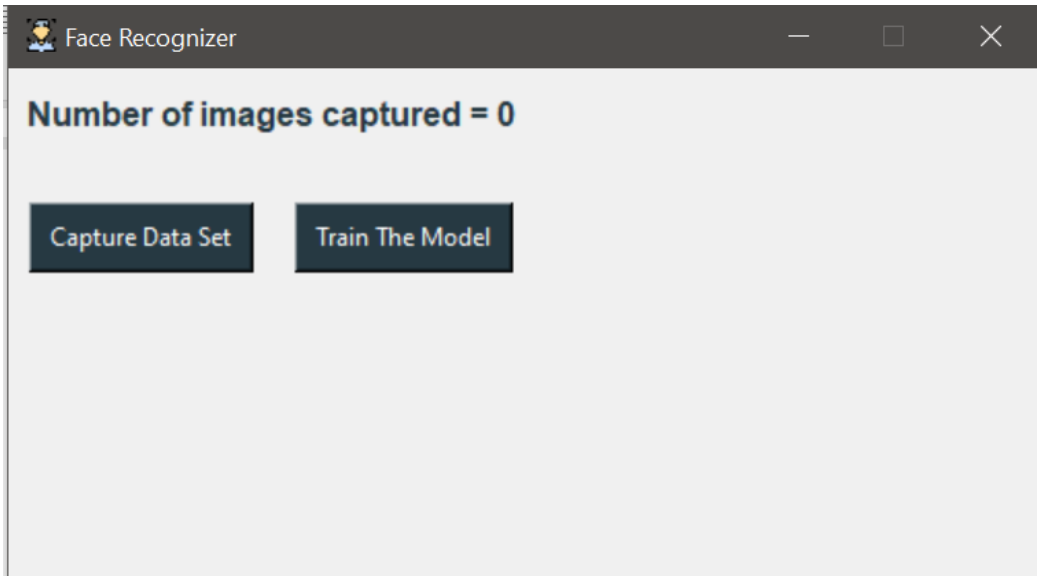

Рисунок В.8 – Вікно функції тренування моделі

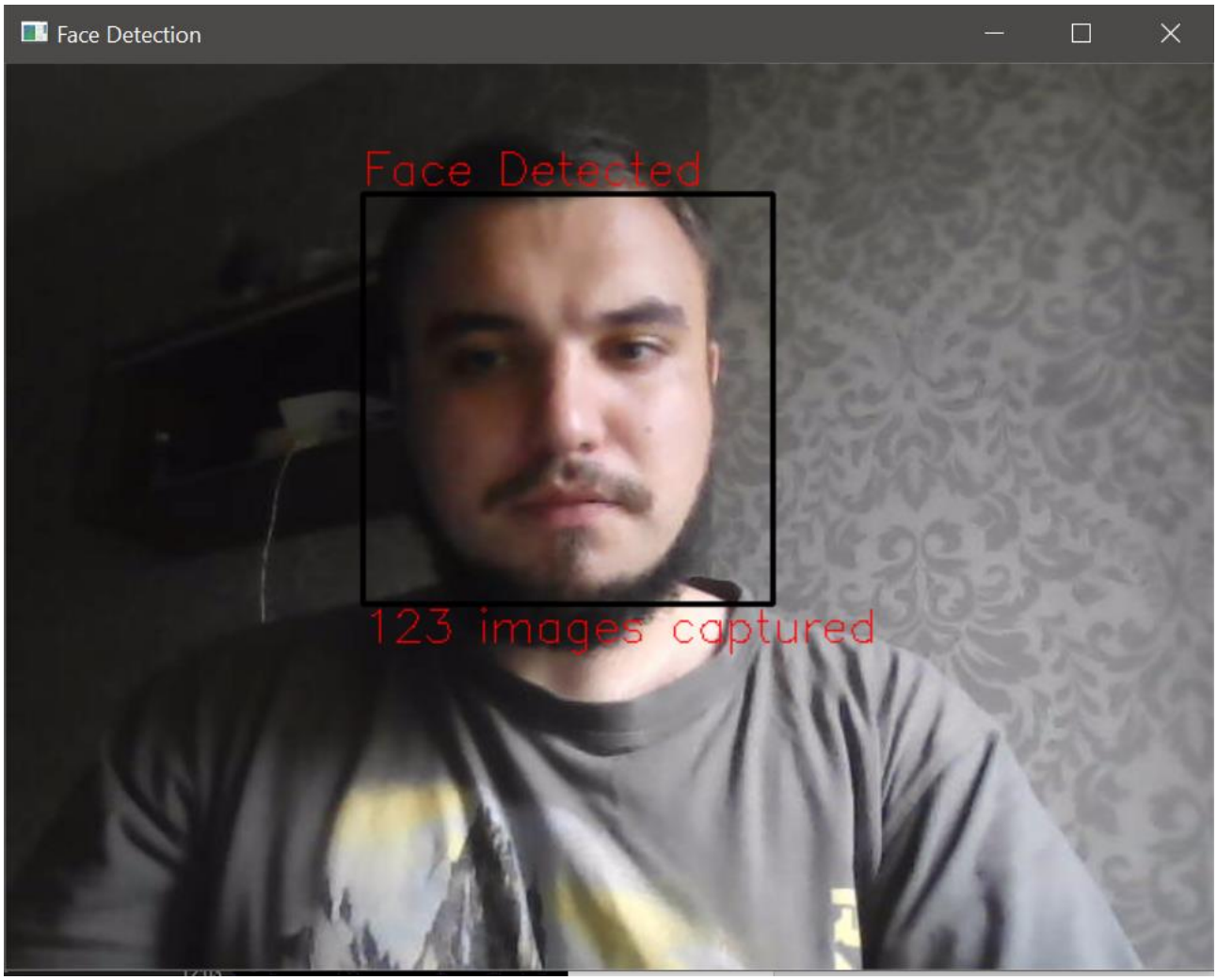

Рисунок В.9 – Процес тренуваня моделі

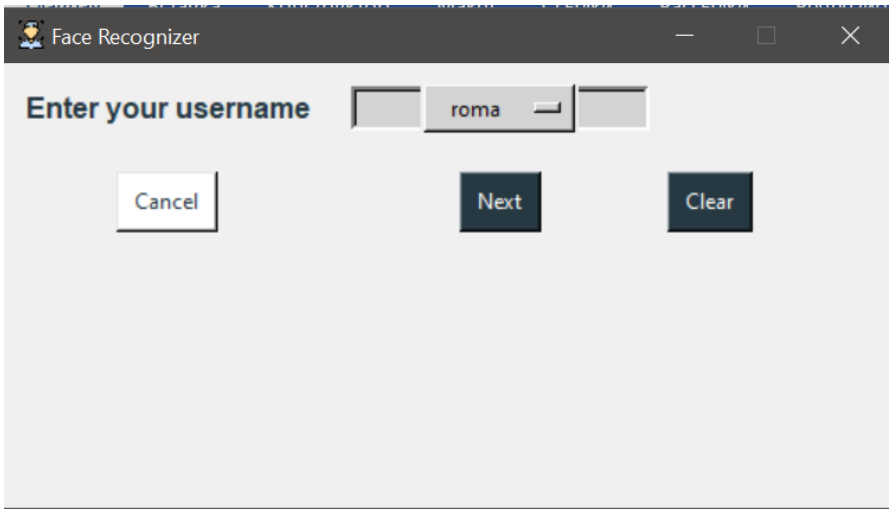

Рисунок В.10 – Вибір користувача

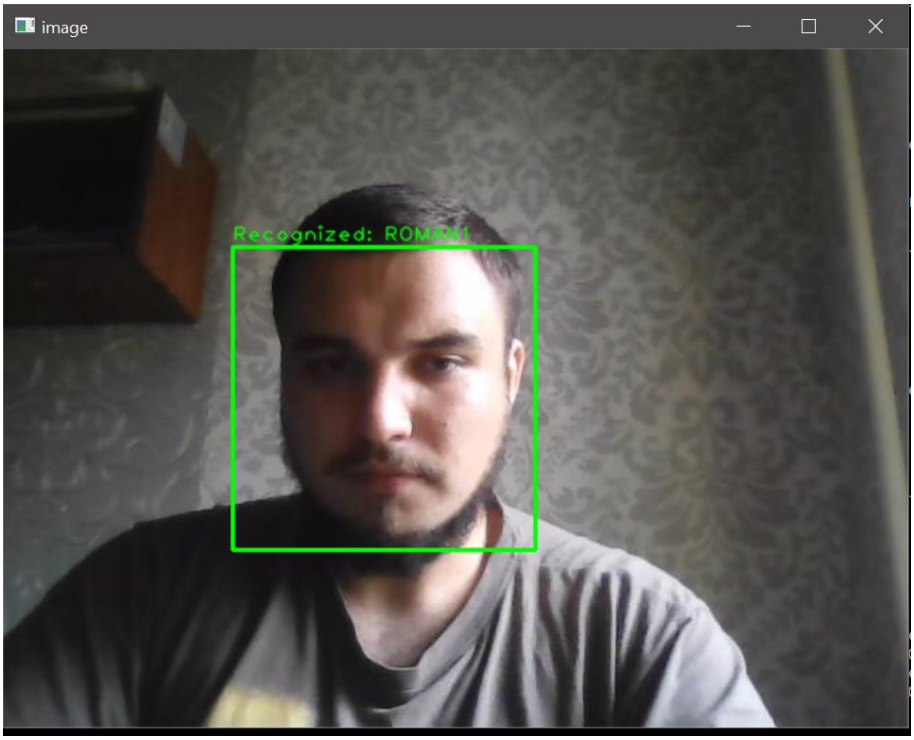

Рисунок В.11 – Ініціалізація обличя по тренованій моделі

ДОДАТОК Г (довідниковий)
Інструкція користувача

Після запуску програми з'являється привітальне вікно (рис. Г.1), яке також містить функцію “Sign In” для початку роботи з модулем.

Рисунок Г.1 – Загальний вид програмного модуля для розпізнавання облич

Після натискання на кнопку «Sign In», користувач переходить на форму ініціалізації, куди вводить своє ім’я. На екрані з'являється поле введення, куди користувачі мають вписати своє ім’я (рис. Г.2).

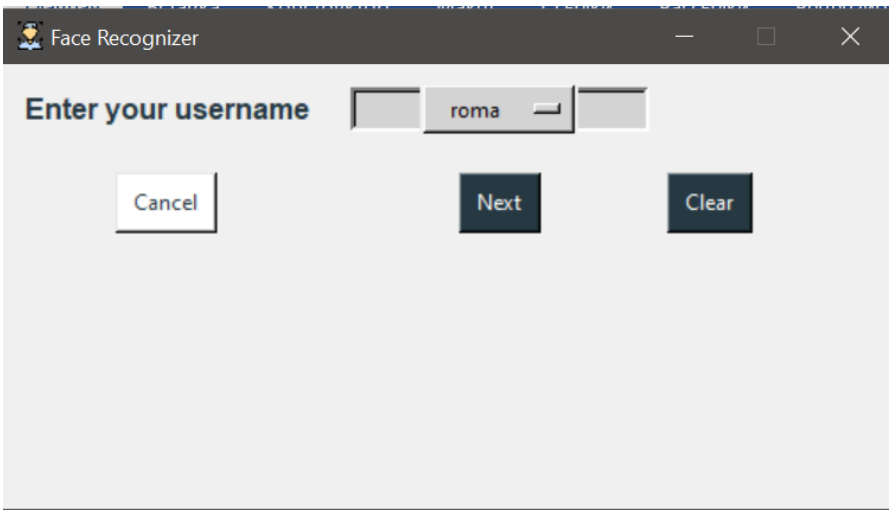

Рисунок Г.2 – Вікно «реєстрації» користувача

При натисненні користувачем на текстове поле, з’являється перелік доступних користувачів (рис. Г.3).

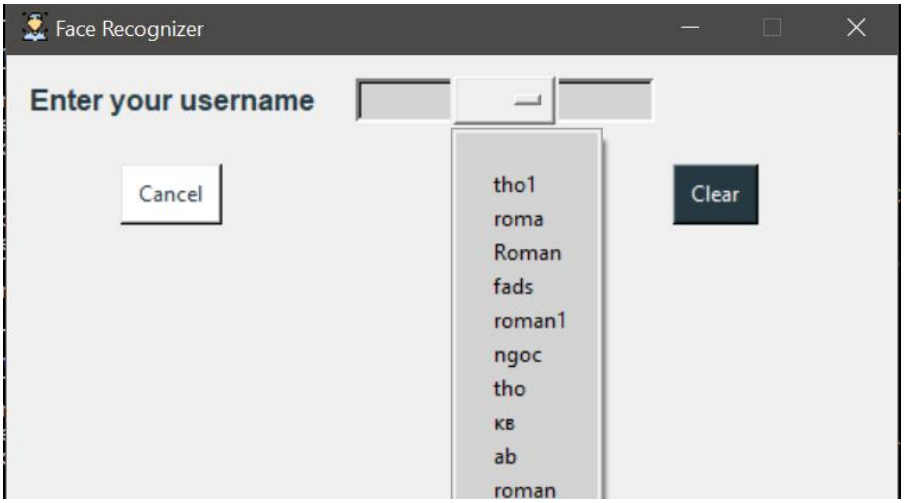

Рисунок Г.3 − Вікно програмного модуля з списком користувачів

При натисканні на кнопку «Next» модуль перекидає користувача на процес розпізнавання, де при правильно натренованій моделі буде ініціалізований (рис. Г.4).

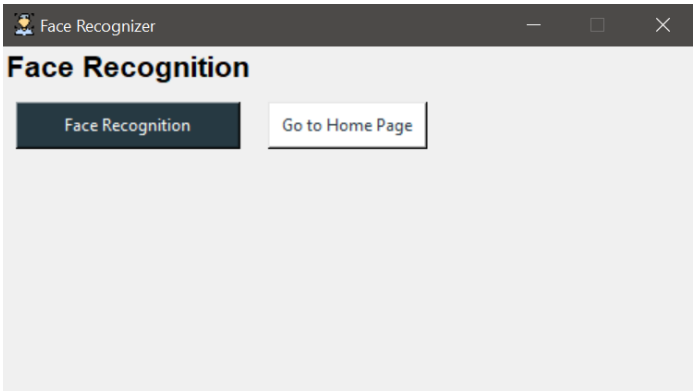

Рисунок Г.4 – Вікно з додатковими функціями у вигляді кнопок

При натисканні на кнопку «Face Recognition» користувач зможе пройти процес розпізнавання (рис. Г.5).

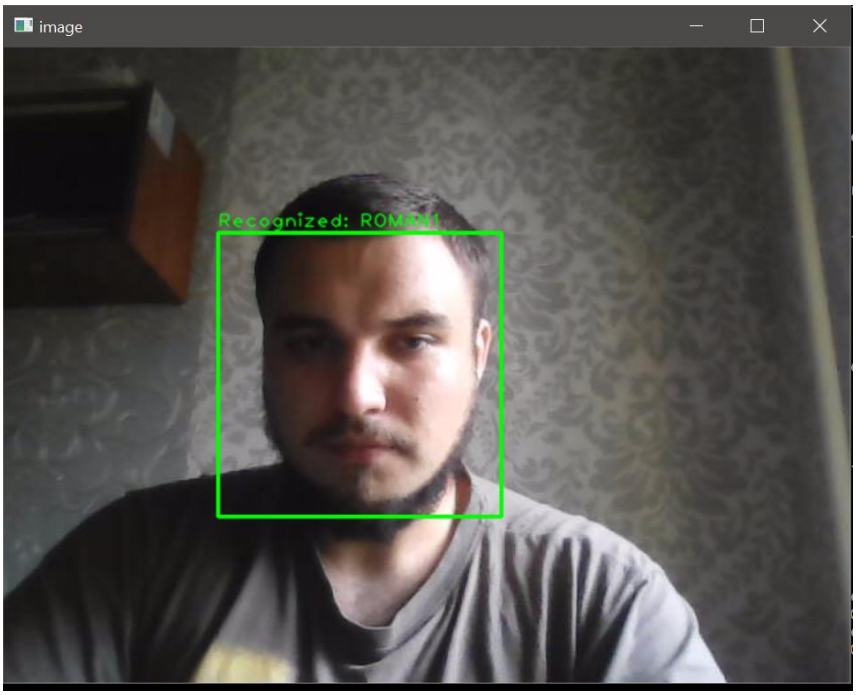

Рисунок Г.5 – Розпізнавання користувача