

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

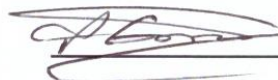
ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект застосунку для підтвердження
особистості на основі біометричних характеристик»

Виконав: студент 4-го курсу групи КБ-21-2
спеціальності 125 «Кібербезпека»

 Артур ЗАГРИВИЙ

Керівник: завідувач кафедри КІ та КБ, к.т.н., доцент

 Андрій ЄФІМЕНКО

Рецензент: доцент кафедри КІ та КБ, к.т.н.,

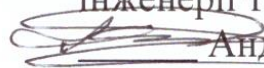
 Олексій ШЕЛУХА

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій ЄФІМЕНКО

5 березня 2025 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Загрий Артур Сергійович

Тема роботи: «Проект застосунку для підтвердження особистості на основі біометричних характеристик»

затверджена Наказом університету від 3 березня 2025 р. №80/с

Термін здачі студентом закінченої роботи 20 червня 2025 р.

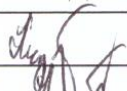
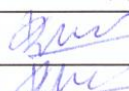
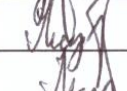
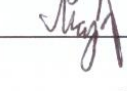
Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

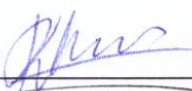
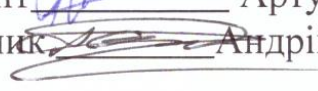
Об'єктом дослідження є процес побудови прототипу застосунку для біометричної ідентифікації особистості за обличчям і голосом. Предметом дослідження є алгоритми розпізнавання обличчя (YOLO, DeepFace) та голосу (SpeechBrain).

Обладнання та програмне забезпечення:

1. Visual Studio Code

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Марина ГРАФ	 03.03.2025	 03.03.2025
2	Марина ГРАФ	 03.03.2025	 03.03.2025
3	Марина ГРАФ	 03.03.2025	 03.03.2025

Студент  Артур ЗАГРИВИЙ
Керівник  Андрій ЄФІМЕНКО

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	03.05.2025	Виконано
2	Аналіз технологій комп'ютерного зору та біометрії	04.03.2025 – 10.03.2025	Виконано
3	Дослідження алгоритмів розпізнавання обличчя (YOLO, DeepFace)	11.03.2025 – 24.03.2025	Виконано
4	Дослідження розпізнавання голосу (SpeechBrain)	25.03.2025 – 01.04.2025	Виконано
5	Розробка модуля виявлення обличчя з камер (OpenCV + YOLO)	02.04.2025 – 15.04.2025	Виконано
6	Розробка модуля розпізнавання обличчя (DeepFace)	16.04.2025 – 26.04.2025	Виконано
7	Розробка модуля розпізнавання голосу (SpeechBrain)	27.04.2025 – 03.05.2025	Виконано
8	Розробка графічного інтерфейсу (PyQt)	04.05.2025 – 11.05.2025	Виконано
9	Інтеграція всіх модулів, тестування та доопрацювання	12.05.2025 – 25.05.2025	Виконано
10	Написання пояснювальної записки	26.05.2025 – 08.06.2025	Виконано
11	Отримання відгуку та рецензії		Виконано
12	Захист	20.06.2025	Виконано

Студент  Артур ЗАГРИВИЙ
Керівник  Андрій ЄФІМЕНКО

АНОТАЦІЯ

Кваліфікаційна робота на тему «Проект застосунку для підтвердження особистості на основі біометричних характеристик (обличчя та голос)» включає вступ, три розділи, висновки, список використаних джерел та додатки.

Текстова частина викладена на 44 сторінках друкованого тексту.

До пояснювальної записки входить список із 19 використаних джерел на 2 сторінках. У роботі представлено 52 рисунки, загальний обсяг – 55 сторінок.

Завданням кваліфікаційної роботи є:

- аналіз сучасних технологій біометричної ідентифікації;
- дослідження алгоритмів розпізнавання обличчя (YOLO, DeepFace);
- дослідження методів розпізнавання голосу (SpeechBrain);
- створення модулів розпізнавання обличчя та голосу з використанням OpenCV, PyQt і відповідних бібліотек;
- інтеграція модулів в єдиний застосунок та демонстрація його роботи.

Перший розділ містить теоретичні відомості щодо біометричної ідентифікації, принципів роботи комп'ютерного зору та обробки мовлення.

У другому розділі проаналізовано та спроектовано основні алгоритми застосунку.

У третьому розділі реалізовано окремі модулі розпізнавання обличчя та голосу та наведено результати тестування, інтеграції, описано роботу застосунку з графічним інтерфейсом.

Ключові слова: БІОМЕТРИЯ, РОЗПІЗНАВАННЯ ОБЛИЧЬ, ГОЛОС, DEEPFACE, SPEECHBRAIN.

					ККІК.КР.Б – 125 – 08 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Проект застосунку для підтвердження особистості на основі біометричних характеристик	Літ.	Арк.	Аркушів
Розроб.		Загривий А.С.		20.06.25				
Керівник		Єфіменко А.А.		20.06.25			4	78
Реценз.		Шелуха О.О.		20.06.25		ФІКТ, Гр. КБ-21-2		
Н. контр.		Граф М.С.		20.06.25				
Затверд.		Єфіменко А.А.		20.06.25				

ANNOTATION

The qualification thesis titled “Application Project for Identity Verification Based on Biometric Characteristics (Face and Voice)” includes an introduction, three chapters, conclusions, a list of references, and appendices.

The textual part consists of 44 pages of printed material.

The explanatory report includes a list of 19 references over 2 pages. The thesis contains 52 figures, with a total volume of 55 pages.

The objectives of the thesis are:

- to analyze modern biometric identification technologies;
- to study face recognition algorithms (YOLO, DeepFace);
- to explore voice recognition methods (SpeechBrain);
- to develop modules for face and voice recognition using OpenCV, PyQt, and relevant libraries;
- to integrate all modules into a unified application and demonstrate its functionality.

The first chapter presents theoretical background on biometric authentication, computer vision, and speech processing.

The second chapter analyzes and designs the key algorithms of the application.

The third chapter implements separate face and voice recognition modules and presents the results of testing, integration, and a description of the application's operation with a graphical user interface.

Keywords: BIOMETRICS, FACE RECOGNITION, VOICE RECOGNITION, DEEP-FACE, SPEECHBRAIN.

		Загрювий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

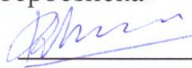
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	7
ВСТУП	9
РОЗДІЛ 1 ТЕОРЕТИЧНИЙ АНАЛІЗ ПРИНЦИПІВ ТА ТЕХНОЛОГІЙ БІОМЕТРИЧНОЇ ІДЕНТИФІКАЦІЇ ТА АУТЕНТИФІКАЦІЇ.....	10
1.1 Аналіз принципів використання біометричних характеристик в задачах кібербезпеки	10
1.2 Аналіз технології розпізнавання обличь DeepFace	14
1.3 Аналіз технології порівняння зразків голосу SpeechBrain	21
1.4 Аналіз технології детекції обличь в відеопотоці You Only Look Once.....	24
1.5 Огляд бібліотеки комп'ютерного бачення OpenCV	30
Висновки до розділу	32
РОЗДІЛ 2 ПРОЄКТУВАННЯ ЗАСТОСУНКУ ДЛЯ ПІДТВЕРДЖЕННЯ ОСОБИСТОСТІ НА ОСНОВІ БІОМЕТРИЧНИХ ХАРАКТЕРИСТИК.....	33
2.1 Розробка загального алгоритму функціонування застосунку	33
2.2 Проєктування бази даних застосунку	37
2.3 Розробка алгоритму ідентифікації обличь	38
2.4 Розробка алгоритму голосової аутентифікації	41
2.5 Проєктування інтерфейсу застосунку.....	43
Висновки до розділу	45
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОТОТИПУ ЗАСТОСУНКУ	46
3.1. Програмна реалізація прототипу застосунку	46
3.2. Тестування роботи застосунку	57
Висновки до розділу	61
ВИСНОВКИ.....	63
СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ.....	65
ДОДАТКИ.....	67

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект застосунку для підтвердження
особистості на основі біометричних характеристик»

Виконав: студент 4-го курсу групи КБ-21-2
спеціальності 125 «Кібербезпека»

 Артур ЗАГРИВИЙ

Керівник: завідувач кафедри КІ та КБ, к.т.н., доцент

 Андрій ЄФІМЕНКО

Рецензент: доцент кафедри КІ та КБ, к.т.н.,

 Олексій ШЕЛУХА

Житомир – 2025

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

		Загривий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

VS Code - Visual Studio Code

YOLO – You Only Look Once

CNN – Convolutional Neural Network

OpenCV – Open-Source Computer Vision Library

DNN – Deep Neural Networks

LFW – Labeled Faces in the Wild

LBPH – Local Binary Patterns Histograms

FAIR – Facebook AI Research

ECAPA-TDNN - Emphasized Channel Attention, Propagation and Aggregation

Time Delay Neural Network

GUI – Graphical User Interface

MFCC – Mel-Frequency Cepstral Coefficients

API – Application Programming Interface

ROI – Region of Interest

RGB – Red Green Blue

BGR – Blue Green Red

ID – Identifier

DB – Database

ML – Machine Learning

OCR – Optical Character Recognition

TTS – Text-to-Speech

STT – Speech-to-Text

CV – Computer Vision

DBMS – Database Management System

SDK – Software Development Kit

		Загрювий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Системи біометричної ідентифікації набувають все більшого поширення в умовах сучасних вимог до інформаційної безпеки, оскільки забезпечують високий рівень точності під час розпізнавання особистості. Одним із ключових напрямів таких технологій є використання аудіовізуальної біометрії, зокрема розпізнавання обличчя та голосу. Завдяки поєднанню цих двох незалежних біометричних характеристик можливо досягти більшої надійності, ніж при використанні лише одного з методів.

Система, що працює в реальному часі, має здатність оперативно здійснювати виявлення особи шляхом захоплення відеопотоку з камери та аудіосигналу з мікрофона, виконувати обробку та порівняння біометричних даних з еталонною базою та генерувати результат аутентифікації.

Метою кваліфікаційної роботи є розробка підсистеми багатофакторної біометричної аутентифікації користувача в реальному часі з використанням розпізнавання обличчя та голосу.

Досягнення поставленої мети передбачає виконання таких завдань:

- дослідити алгоритми та моделі для розпізнавання обличчя (YOLO, DeepFace) та голосу (SpeechBrain);
- реалізувати інтеграцію системи в користувацький графічний інтерфейс;
- провести тестування точності, швидкодії та надійності системи в умовах реального часу.

Об'єктом дослідження є процес побудови прототипу застосунку для біометричної ідентифікації особистості за обличчям і голосом

Предметом дослідження є алгоритми розпізнавання обличчя (YOLO, DeepFace) та голосу (SpeechBrain).

У процесі виконання кваліфікаційної роботи було створено програмну підсистему, що реалізує багатофакторну біометричну перевірку особи шляхом поєднання розпізнавання обличчя та голосу. Система дозволяє ідентифікувати користувача, забезпечуючи базовий рівень захисту доступу до обчислювальних ресурсів або критичних програмних компонентів.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 ТЕОРЕТИЧНИЙ АНАЛІЗ ПРИНЦИПІВ ТА ТЕХНОЛОГІЙ БІОМЕТРИЧНОЇ ІДЕНТИФІКАЦІЇ ТА АУТЕНТИФІКАЦІЇ

1.1 Аналіз принципів використання біометричних характеристик в задачах Кібербезпеки

Із давніх-давен люди впізнавали одне одного за рисами обличчя, голосом та іншими ознаками, притаманними лише конкретній особі. Навіть сьогодні обличчя є основною ознакою, за якою здійснюється ідентифікація особистості — під час перевірки паспорта, водійських прав чи перепустки на територію організації обов’язково звіряють фотографію з обличчям людини, яка надає документ.

Сучасні технології здатні забезпечити достовірне підтвердження особи за допомогою унікальних для неї характеристик. Такі технології базуються на досягненнях біометрики (або біометрії) — наукової дисципліни, що займається статистичним аналізом біологічних спостережень і явищ.

Біометрична характеристика — це вимірювана фізіологічна або поведінкова ознака живої людини, яку можна використати для встановлення особи або перевірки задекларованих персональних даних.

Оскільки біометричний параметр є унікальним для конкретної людини, його можна застосовувати для однофакторної автентифікації. Також його можна комбінувати з паролем чи пристроєм автентифікації (наприклад, смарт-картою) для забезпечення двофакторного захисту.

Біометрична автентифікація зазвичай є одним із найзручніших способів підтвердження особи для користувача. У більшості випадків добре спроектована біометрична система зчитує показники з людини та правильно виконує процедуру автентифікації.

Фізіологічні біометричні характеристики (фізичні або статичні біометричні характеристики) — це біометричні ознаки, що базуються на вимірюванні анатомічних особливостей людини.

До фізіологічних біометричних характеристик належать:

— райдужна оболонка ока;

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

- відбитки пальців;
- долоня;
- сітківка ока.

Поведінкові біометричні характеристики. Характерною особливістю поведінкових параметрів є їхня розтягнутість у часі: вимірювана дія має початок, середину та завершення.

До поведінкових біометричних характеристик належать:

- голос;
- підпис;
- ритми роботи серця.

Поведінкові біометричні характеристики (динамічні біометричні характеристики) — це біометричні ознаки, що ґрунтуються на даних, отриманих шляхом вимірювання дій або звичок людини.

Розмежування між поведінковими та фізіологічними характеристиками є досить умовним. Поведінкові параметри значною мірою залежать від фізіологічних особливостей. Наприклад, голос залежить від форми голосових зв'язок, а підпис — від спритності кисті та пальців. Деякі фізіологічні характеристики, як-от обличчя, можуть змінюватися з віком або поведінкою людини. Водночас і поведінка — наприклад, як саме людина прикладає палець або дивиться в камеру — може впливати на ефективність роботи біометричної системи.

Фізіологічні біометричні характеристики зазвичай залишаються незмінними протягом усього життя. Їх використання для автентифікації часто сприймається як примус чи втручання в приватне життя. Поведінкові ж характеристики сприймаються м'якше, але вони менш стабільні: можуть змінюватися під впливом стресу, хвороб і загалом забезпечують нижчу якість автентифікації порівняно з фізіологічними параметрами.

Робота біометричних систем. Хоча біометричні технології відрізняються за об'єктами й методами вимірювання, усі біометричні системи працюють за єдиним принципом(рис.1.1). Користувач надає зразок — сире зображення або запис

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

фізіологічної чи поведінкової характеристики. За допомогою реєструвального пристрою (наприклад, сканера або камери) цей зразок обробляється для виокремлення унікальних ознак, в результаті чого формується контрольний шаблон. Цей шаблон представляє собою числову послідовність, з якої неможливо відновити первинний зразок. Контрольний шаблон фактично є своєрідним «паролем» користувача.

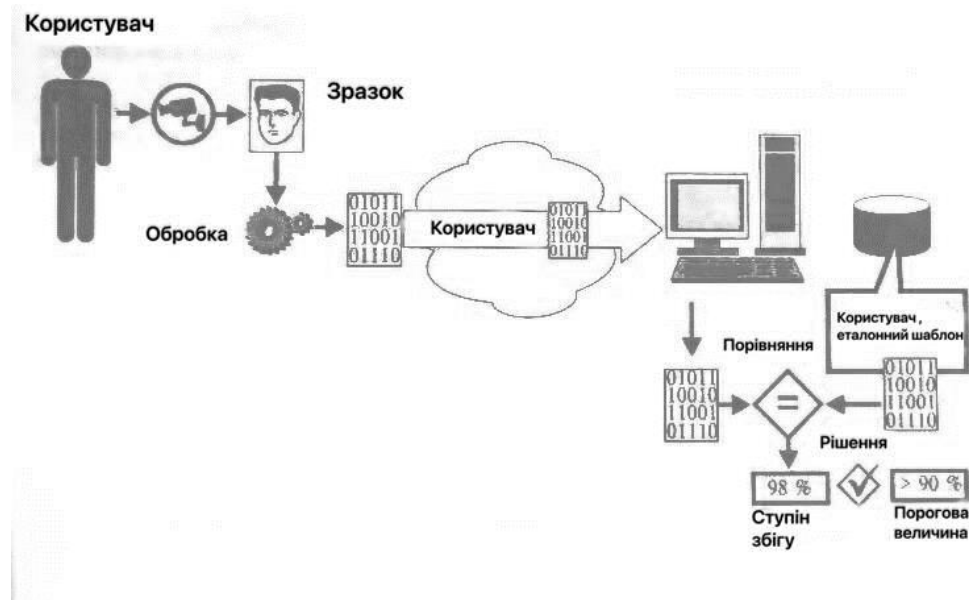


Рисунок 1.1 — Принцип роботи аутентифікації на основі біометричних показників

Контрольний шаблон порівнюється з еталонним шаблоном, створеним під час реєстрації користувача на основі кількох зразків його біометричної характеристики. Оскільки шаблони ніколи не збігаються повністю, система приймає рішення про те, наскільки збіг є «достатнім». Цей рівень збігу має перевищувати заздалегідь визначене порогове значення.

Біометричні системи не є безпомилковими:

- контрольний шаблон може бути помилково співвіднесений з еталонним шаблоном іншої особи;
- або, навпаки, не буде визнаний як відповідний навіть справжньому користувачу.

Оцінка точності біометричних систем. Точність роботи біометричної системи визначається двома показниками:

- Коефіцієнт хибних збігів (FMR) — також відомий як помилка першого роду або ймовірність помилкового допуску (FAR);

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

- Коефіцієнт хибних незбігів (FNMR) — також називається помилкою другого роду або ймовірністю відмови в доступі справжньому користувачеві (FRR).

Системи з низьким значенням FMR є більш захищеними, а системи з низьким FNMR — зручнішими у використанні. Зазвичай ці два параметри перебувають у протиріччі: підвищення безпеки часто знижує зручність використання.

Якість реєстрації та шаблону є критичними факторами загальної ефективності системи. Якщо шаблон сформовано неякісно, користувачу може знадобитися повторна реєстрація.

Режими роботи біометричних систем. У режимі автентифікації система перевіряє заявлену особу — тобто порівнює контрольний шаблон зі збереженим еталонним шаблоном (режим 1:1). Особу попередньо зазначають, наприклад, за допомогою введення імені. У деяких системах дозволено вибір із кількох шаблонів: наприклад, користувач із трьома зареєстрованими відбитками пальців може прикласти будь-який із трьох.

У режимі розпізнавання система встановлює особу без попереднього введення імені — контрольний шаблон порівнюється з великою кількістю еталонних шаблонів у базі (режим 1:N). Якщо виявлено збіг, особу одразу ідентифікують і підтверджують.

Також існує проміжний варіант — режим 1:few, коли система здійснює пошук серед невеликої групи зареєстрованих користувачів. Якщо база перевищує 500 записів — система вважається повноцінною 1:N.

Недоліки біометричної автентифікації.

Втручання в приватність. Багатьох користувачів, особливо клієнтів, турбує питання зберігання та поширення їхніх біометричних даних. Якщо в організації централізоване сховище, а користувачі не можуть контролювати доступ до своїх даних — виникає ризик:

- зловживань (наприклад, незаконний обмін даними);
- нецільового використання (наприклад, застосування для інших цілей без згоди).

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Особисті, культурні та релігійні аспекти. Наприклад, дактилоскопічні системи можуть асоціюватися в користувачів із криміналістикою.

Гігієна та безпека. Виникають питання: чи обробляють пристрій антисептиком після кожного дотику? Чи не є він травмонебезпечним? Наприклад, при скануванні сітківки використовується світло, направлене в око.

Непридатність для всіх. Від 1 до 3% людей фізично не можуть використовувати деякі біометричні системи: німі не зможуть пройти голосову автентифікацію; люди без пальців або кистей не зможуть користуватись системами, що зчитують відбитки.

1.2 Аналіз технології розпізнавання обличь DeepFace

DeepFace(рис.1.2.) — це система розпізнавання облич, розроблена в 2014 році дослідницькою групою Facebook AI Research (FAIR). Її головна мета — точно розпізнавати обличчя незалежно від положення голови, освітлення, виразу та інших варіацій. DeepFace став проривом у своєму часі, оскільки продемонстрував людський рівень точності (97,35%) на стандартному наборі LFW[7]. Для свого часу це була одна з перших глибоких нейронних мереж, яка показала можливість побудови узагальненої системи розпізнавання облич із високою точністю на великих реальних даних.

Ключовим у DeepFace є використання глибокої згорткової нейронної мережі, натренованої на великому наборі з понад 4 мільйонів зображень приблизно 4 тисяч різних осіб. Але ще до подачі зображення в мережу важливу роль відіграє етап попередньої обробки — так званої нормалізації обличчя. Система застосовує 3D-модель голови для вирівнювання обличчя таким чином, щоб воно дивилося прямо в камеру, незалежно від початкового кута повороту(рис.1.2.). Це вирівнювання критично важливе, бо дозволяє зробити вхідні зображення більш подібними між собою, що значно спрощує роботу нейронної мережі.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

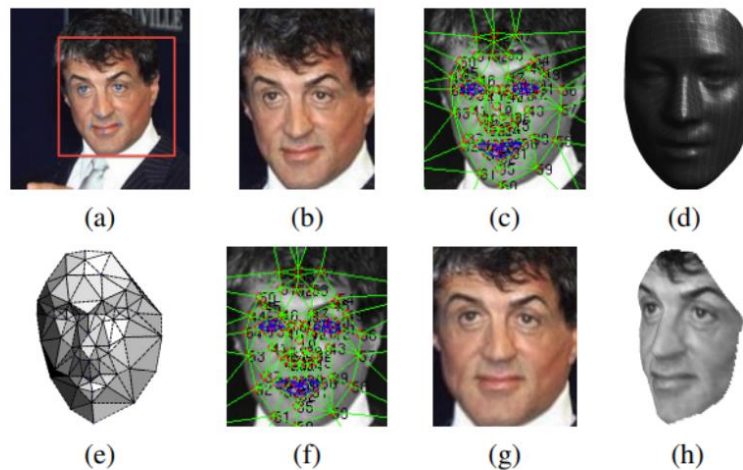


Рисунок 1.2. Алгоритм вирівнювання обличчя

Після вирівнювання зображення масштабується до розміру 152×152 пікселів і передається у глибоку нейронну мережу, що складається з дев'яти шарів, з яких сім — згорткові. На перших шарах витягуються базові патерни — краї, контури, текстури, а на більш глибоких — ознаки високого рівня, такі як форми очей, носа, губ, особливості лоба тощо. На останньому повнозв'язному шарі мережа формує вектор-репрезентацію обличчя — числовий вектор фіксованої довжини, який описує людину. У DeerFace цей вектор має розмір 4096 компонентів.

Суть розпізнавання полягає в тому, що обличчя різних людей повинні мати якомога віддаленіші вектори у цьому багатовимірному просторі, тоді як зображення одного і того ж обличчя — навпаки, повинні бути дуже близько. Для навчання мережі застосовується метод парного навчання: мережа порівнює два обличчя (які можуть бути або з однієї людини, або з різних), і вчиться мінімізувати відстань між векторами для одного обличчя і максимізувати для різних. Застосовується так звана loss-функція на основі евклідової відстані (або пізніше cosine similarity), яка оптимізує простір репрезентацій.

Після навчання мережа формує embedding-простір, у якому обличчя кожної особи сконцентровані в локальному кластері, а кластери різних осіб добре розділені. Цей простір є майже евклідовим і підпорядковується метриці косинусної подібності або евклідової відстані. Процес ідентифікації особи зводиться до дуже

швидкої операції. Зображення нового обличчя передається у модель, з нього витягується вектор ознак, який порівнюється з векторами із бази даних. Якщо найближча відстань до відомого обличчя нижча за певний поріг, то система приймає, що це та сама людина. Таким чином, DeepFace не просто класифікує людей, а створює уніфіковану систему для векторного пошуку у великому наборі[4].

Функціонально DeepFace реалізує механізм:

$$\text{identity}(I) = \arg \min_k \|f(I) - \mu_k\|_2$$

де μ_k — центр кластера k -ої особи у базі даних.

Таким чином, DeepFace розв'язує задачу face recognition як класифікацію у багатовимірному латентному просторі.

Істотною особливістю DeepFace є те, що вона була однією з перших систем, яка довела практичність використання end-to-end глибокого навчання для задач біометричної ідентифікації. На відміну від класичних підходів (наприклад, LBPН, Eigenfaces, Fisherfaces), які потребували ручного виділення ознак, DeepFace автоматично навчалася витягувати ознаки, релевантні для ідентифікації. Цей підхід зробив систему стійкою до змін зовнішнього вигляду, освітлення, часткового закриття обличчя.

DeepFace також задав напрямок для розвитку подальших систем, таких як DeepID, FaceNet, VGGFace, ArcFace, CosFace. Багато з них застосовують подібну ідею — перетворення зображення обличчя у числовий вектор та оптимізацію його положення у просторі в залежності від міжособових відстаней. У сучасних системах використовуються вдосконалені loss-функції, як-от Triplet loss, Additive Margin Softmax, ArcFace loss, які ще краще структурують простір репрезентацій.

У прикладному сенсі DeepFace і її наступники лежать в основі багатьох систем розпізнавання облич, включаючи безпеку пристроїв, відеоспостереження, банківські сервіси, системи перевірки ідентичності, розпізнавання емоцій і навіть демографічний аналіз. Їх можна реалізувати за допомогою відкритих бібліотек, таких як deepface — Python-пакет, який обгортає кілька моделей (включно з VGGFace,

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Facenet, ArcFace) і дозволяє легко використовувати розпізнавання, верифікацію, виявлення віку, статі, емоцій.

В проєкті застосунку використовується поєднання технології DeepFace в поєднанні з похідною технологією FaceNet.

FaceNet — це одна з найвідоміших та найефективніших глибоких нейронних мереж, призначених для розпізнавання облич, яка була запропонована дослідниками Google у 2015 році.

Алгоритм FaceNet ґрунтується на глибокій згортковій нейронній мережі, метою якої є трансформація вхідного зображення обличчя у вектор ознак фіксованої довжини — так званий ембедінг. Цей ембедінг є компактним числовим представленням обличчя у багатовимірному просторі, яке зберігає інформацію про індивідуальні риси особи та дозволяє проводити ефективне порівняння між різними обличчями.

Ключовим елементом алгоритму є функція втрат triplet loss(рис.1.3), яка визначає, як саме навчається мережа. На кожному кроці навчання мережа працює з трійкою зображень: опорним (anchor), позитивним (positive) і негативним (negative). Anchor та positive зображення належать тій самій особі, тоді як negative — іншій. Метою є навчити мережу таким чином, щоб відстань між векторами ознак anchor і positive була меншою за відстань між anchor і negative принаймні на деякий заздалегідь визначений поріг (margin). Таким чином забезпечується правильне розташування облич у векторному просторі, де особи згруповані відповідно до схожості [12].

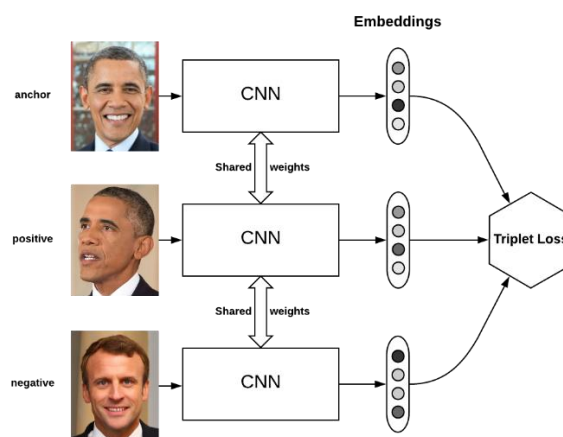


Рисунок 1.3 — Алгоритм вирівнювання обличчя

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

Формально triplet loss визначається як:

$$\mathcal{L} = \sum_{i=1}^N [\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha]_+$$

де $f(x)$ — це ембедінг зображення;

x_i^a x_i^p x_i^n — anchor, positive і negative відповідно;

α — це margin, що визначає мінімальний бажаний розрив між відстанями.

Для побудови самої моделі FaceNet можуть використовуватись різні архітектури нейронних мереж. Найчастіше застосовуються модифіковані Inception або Inception-ResNet, які демонструють хороше співвідношення між точністю та обчислювальними витратами. Перед подачею зображення до мережі воно проходить стадії попередньої обробки, зокрема виявлення та вирівнювання обличчя (наприклад, за допомогою MTCNN або dlib), що суттєво покращує точність.

Після проходження через мережу кожне зображення отримує вектор ембедінгу довжиною, як правило, 128. Цей вектор вже можна використовувати для вирішення задачі верифікації (чи належать два зображення одній особі) або ідентифікації (знаходження особи серед багатьох).

У практичних застосуваннях, таких як бібліотека DeepFace, алгоритм FaceNet використовується в оптимізованому вигляді — ембедінги порівнюються через евклідову відстань або косинусну подібність. Наприклад, якщо відстань між ембедінгами двох зображень є меншою за певний поріг, вважається, що на них зображена одна й та сама особа. Цей підхід дозволяє ефективно вирішувати задачу розпізнавання облич у реальному часі навіть на пристроях з обмеженими ресурсами.

FaceNet набув широкого застосування в біометричних системах завдяки високій точності, яку модель продемонструвала на таких відомих наборах даних, як LFW, YouTube Faces DB та ін. Проте пряме використання оригінальної реалізації FaceNet є досить складним, оскільки вимагає глибоких знань у галузі машинного навчання та потужних обчислювальних ресурсів.

Інтеграція FaceNet у DeepFace дозволяє поєднати точність і надійність моделі з простотою її використання. Це відкриває можливість створення систем реального часу для розпізнавання облич без необхідності самостійного тренування моделі або

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

складної конфігурації. З технічної точки зору, така інтеграція полягає у використанні Keras або TensorFlow-реалізацій моделі FaceNet, які попередньо завантажуються та використовуються всередині високорівневого API DeepFace.

FaceNet залишається однією з ключових технологій у сфері комп'ютерного зору для біометричної аутентифікації, а її доступність через бібліотеки на кшталт DeepFace значно розширює можливості її практичного застосування в наукових дослідженнях та прикладних проєктах. У контексті створення системи аутентифікації користувача за допомогою розпізнавання облич, використання FaceNet забезпечує високу точність, узагальнюваність і масштабованість рішення.

Для використання DeepFace та FaceNet необхідне використання платформи TensorFlow та Keras з їх потужними функціоналами та можливостями. Також необхідне поєднання з бібліотеками OpenCV та Dlib (рис.1.4).

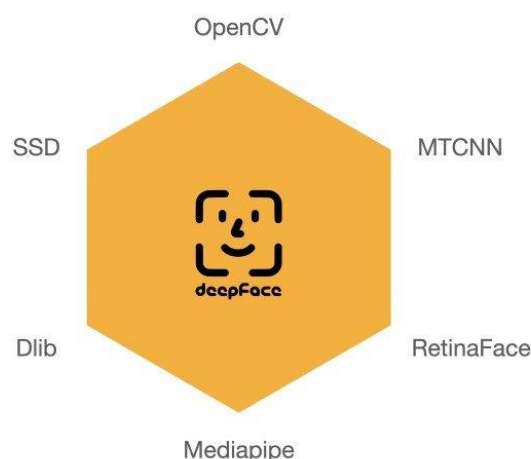


Рисунок 1.4 — Сукупність технологій, на яких працює DeepFace

TensorFlow — це потужна платформа з відкритим кодом для реалізації та розгортання моделей машинного навчання і глибокого навчання, яка була розроблена дослідниками компанії Google та вперше представлена широкому загалу у 2015 році. Вона стала одним із найбільш популярних інструментів для побудови, тренування та масштабування штучних нейронних мереж завдяки своїй гнучкості, модульності та активній підтримці спільноти.

		Загивий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

Архітектурно TensorFlow побудований як система для обчислень із графами залежностей. У ній усі математичні операції, включно з обробкою тензорів (багатовимірних масивів), виконуються у вигляді вузлів обчислювального графа, що дозволяє ефективно розпаралелювати виконання на різних апаратних платформах — центральних (CPU) і графічних (GPU) процесорах, а також на спеціалізованих чипах, таких як TPU. Це забезпечує масштабованість моделей від рівня експериментальних розробок до промислового використання на кластерних інфраструктурах.

Однією з головних переваг TensorFlow є підтримка високорівневого API — Keras, який дозволяє описувати архітектуру нейронних мереж у вигляді зрозумілих та компактних структур. Такий підхід істотно знижує поріг входження для дослідників і розробників, а також полегшує розгортання вже існуючих моделей у вигляді мікросервісів або інтегрованих модулів.

У задачах комп'ютерного зору, зокрема розпізнавання облич, TensorFlow широко використовується як платформа для тренування та використання моделей згорткових нейронних мереж. Наприклад, модель FaceNet, яка виконує перетворення зображення обличчя у вектор ембедінгу, реалізована у TensorFlow як у вигляді офіційних імплементацій, так і численних адаптацій у відкритому доступі. Ці моделі можуть бути легко завантажені, відновлені з чекпойнтів, використані для інференсу (прогнозування), а також модифіковані для подальшого донавчання на кастомізованих датасетах.

TensorFlow також підтримує функціональність для попередньої обробки даних, включаючи масштабування зображень, вирівнювання, нормалізацію та генерацію батчів, що є критично важливим при роботі з великими наборами зображень облич. Крім того, бібліотека пропонує інструменти для візуалізації навчального процесу (наприклад, TensorBoard), оптимізації обчислень, збереження моделей у форматі SavedModel, а також конвертації їх у формати для мобільних або вебплатформ (TensorFlow Lite, TensorFlow.js).

Важливо зазначити, що TensorFlow не обмежується лише підтримкою згорткових мереж. Він також включає функціональність для рекурентних нейронних

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

мереж, трансформерів, автокодерів та інших сучасних архітектур. Завдяки цьому TensorFlow часто обирається як єдина платформа для реалізації повного циклу розробки інтелектуальних систем — від попередньої обробки даних і тренування моделей до їх інтеграції в готові програмні продукти.

TensorFlow виступає як ключова технологічна платформа для створення, навчання та експлуатації моделей глибокого навчання. У контексті системи біометричної аутентифікації на основі розпізнавання облич TensorFlow забезпечує надійну базу для реалізації алгоритмів, таких як FaceNet, дозволяючи поєднувати високу точність, масштабованість і гнучкість у налаштуванні.

1.3 Аналіз технології порівняння зразків голосу SpeechBrain

SpeechBrain — це потужна платформа з відкритим вихідним кодом для побудови, тренування та застосування моделей обробки голосу і мовлення. Проєкт був започаткований у 2021 році групою дослідників з метою створення універсальної та модульної екосистеми для досліджень і впровадження алгоритмів у сфері автоматичного розпізнавання мовлення, верифікації мовців, синтезу мовлення, аналізу емоцій, сегментації голосу тощо[9].

В основі SpeechBrain лежить фреймворк PyTorch, що забезпечує високу гнучкість при проєктуванні нейронних архітектур та можливість інтеграції з сучасними підходами глибокого навчання[6]. SpeechBrain організовано за принципом конфігурованих сценаріїв (YAML-конфігурації), що дозволяє швидко змінювати параметри навчання, архітектури моделей, алгоритми обробки сигналів і стратегії оптимізації без необхідності втручання в код основних компонентів.

Однією з найпопулярніших можливостей, які надає SpeechBrain, є верифікація мовця — процедура, що дозволяє підтвердити або спростувати належність голосового зразка конкретній особі. У цьому контексті бібліотека пропонує готову до використання попередньо натреновану модель ECAPA-TDNN, яка демонструє високу точність навіть при наявності шумів, варіацій мови чи емоційного

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

стану. Ця модель перетворює аудіосигнал у вектор ознак, який надалі використовується для порівняння з іншими векторами на основі косинусної подібності або евклідової відстані.

Алгоритмічна реалізація системи включає як попередню обробку сигналу, так і глибоку нейронну архітектуру для побудови ембедінгів. У ході навчання система оптимізується за допомогою втрат типу Additive Angular Margin Loss (Рис. 1.5), яка сприяє чіткішому розділенню ембедінгів мовців у просторовому векторному просторі.

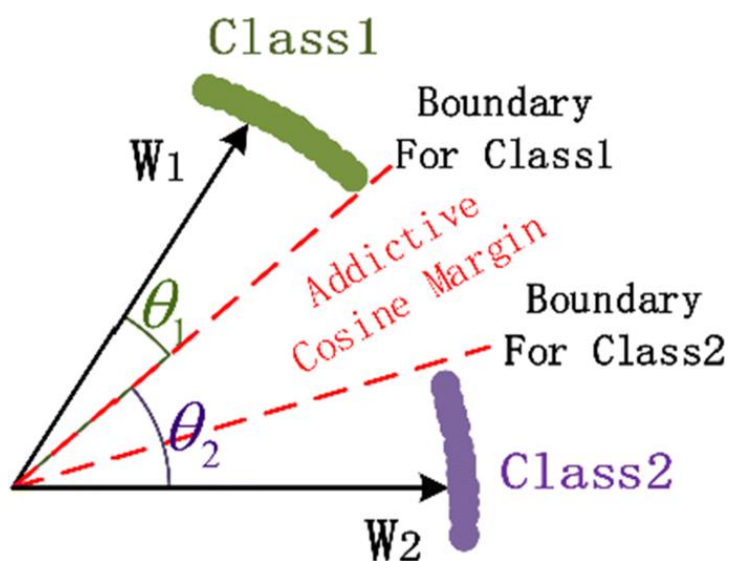


Рисунок 1.5 — Additive Angular Margin Loss

Алгоритм можна умовно поділити на кілька етапів: попередня обробка аудіосигналу, вилучення ознак, генерація ембедінгів, нормалізація та зіставлення. Розглянемо ці етапи детальніше.

Попередня обробка сигналу. На першому етапі обробляється аудіосигнал мовця, який зазвичай є монофонічним сигналом із частотою дискретизації 16 кГц. Перед подачею в нейронну мережу здійснюється нормалізація гучності та обтинання тиші, що дозволяє сконцентрувати увагу моделі лише на мовному фрагменті без шумових артефактів. Опціонально може виконуватись перетворення сигналу в спектральні представлення.

Вилучення ознак (Feature Extraction). На цьому етапі з аудіосигналу витягуються низькорівневі ознаки, які є вхідними для нейронної мережі. У випадку

ESCAPA-TDNN зазвичай використовується набір фільтрбанкових ознак, що формують послідовність спектральних векторів у часі. Ці ознаки дозволяють моделі вловлювати частотні характеристики мовлення, що є критичними для розрізнення голосів.

Побудова ембедінгу голосу. Основна частина алгоритму полягає у проходженні ознак через модель ESCAPA-TDNN. Ця архітектура складається з кількох TDNN-блоків із residual-з'єднаннями, механізмами уваги по каналах (SE блоки), а також з шаром Global Context Aggregation, який дозволяє мережі враховувати інформацію про довготривалі залежності. У результаті проходження сигналу через мережу формується вектор фіксованої довжини — ембедінг мовця. Цей вектор відображає унікальні характеристики голосу: тембр, інтонацію, форму мовленнєвого апарату, манеру вимови тощо.

Нормалізація ембедінгів. З метою забезпечення стабільності метрик відстані ембедінги нормалізуються за допомогою L2-норми. Таким чином, кожен ембедінг представлений як точка на гіперсфері у векторному просторі. Це дозволяє ефективно застосовувати косинусну відстань або подібність між векторами для задач класифікації або верифікації.

Порівняння ембедінгів. Верифікація здійснюється шляхом обчислення косинусної подібності між ембедінгом референтного зразка та ембедінгом поточного запису.

Формула косинусної подібності між двома векторами A та B має вигляд:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \cdot \|\mathbf{B}\|}$$

де: $\mathbf{A} \cdot \mathbf{B}$ — скалярний добуток векторів A і B;

$\|\mathbf{A}\|$ — евклідова норма (довжина) вектора A;

$\|\mathbf{B}\|$ — евклідова норма (довжина) вектора B;

θ — кут між векторами у векторному просторі.

Якщо значення подібності перевищує певний поріг, вважається, що зразки належать одному мовцю. У системах на кшталт SpeechBrain цей поріг може бути динамічним або визначеним емпірично в процесі калібрування системи.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

SpeechBrain підтримує як offline, так і online режими роботи, забезпечуючи можливість реалізації реального часу для задач розпізнавання або ідентифікації голосу. Це особливо важливо для інтерактивних додатків, зокрема біометричних систем доступу, голосових асистентів або моніторингових систем, де швидкість прийняття рішень є критичною.

Інтеграція з іншими бібліотеками, такими як torchaudio, soundfile та librosa, дозволяє розширювати можливості системи обробки аудіо, уникаючи дублювання функціональності. Завдяки повній відкритості коду та активній спільноті, SpeechBrain надає можливості як для дослідницьких проєктів, так і для розгортання у промислових масштабах.

У контексті розробки системи голосової верифікації користувача SpeechBrain є однією з найбільш зручних та ефективних платформ[13]. Вона дозволяє без значних зусиль реалізувати процедуру запису аудіо, обробки сигналу, побудови ембедінгів і порівняння голосів, що робить її ідеальним вибором для систем безпеки, де важлива точність, масштабованість та адаптивність до нових умов.

1.4 Аналіз технології детекції обличч в відеопотоці You Only Look Once

YOLO — це один із найвідоміших та найефективніших алгоритмів для виявлення об'єктів на зображеннях і відео в реальному часі. Його основна перевага — швидкість і точність. YOLO підходить для застосунків, де важлива обробка у реальному часі, таких як відеоспостереження, автономні транспортні засоби, дрони та розумні системи безпеки.

YOLO — це одностадійний детектор. Це означає, що він виконує виявлення об'єктів у єдиному проході через нейронну мережу, на відміну від дворівневих підходів, таких як R-CNN, які спочатку генерують регіони-пропозиції, а потім класифікують їх[5]. У YOLO зображення розбивається на сітку, і кожна комірка сітки відповідає за виявлення об'єктів, центри яких потрапляють у цю комірку. Для кожної комірки модель прогнозує обмежувальні рамки “bounding boxes”, ймовірності класів об'єктів та рівень впевненості “confidence score”.

		Загивий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

Перші версії YOLO були побудовані на основі звичайних згорткових нейронних мереж. YOLOv1 мав певні обмеження у точності, особливо для маленьких об'єктів, однак уже тоді демонстрував вражаючу швидкість. YOLOv2 (також відомий як YOLO9000) покращив точність за рахунок використання покращених архітектур, як-от Batch Normalization та використання попередньо натренованої нейромережі Darknet-19. YOLOv3 став ще більш потужним завдяки використанню мережі Darknet-53, а також механізму виявлення об'єктів на кількох масштабах, що значно покращило роботу з малими об'єктами.

З приходом YOLOv4 і далі до YOLOv5, YOLOv6 і YOLOv8, були впроваджені нові техніки з глибокого навчання. Наприклад, у YOLOv4 застосовуються такі методики, як Mosaic Data Augmentation, Cross-Stage Partial Networks (CSPNet), DropBlock та самонавчання. Всі ці підходи спрямовані на покращення узагальнення, стабільності тренування та підвищення продуктивності без жертвування швидкістю (рис.1.6).

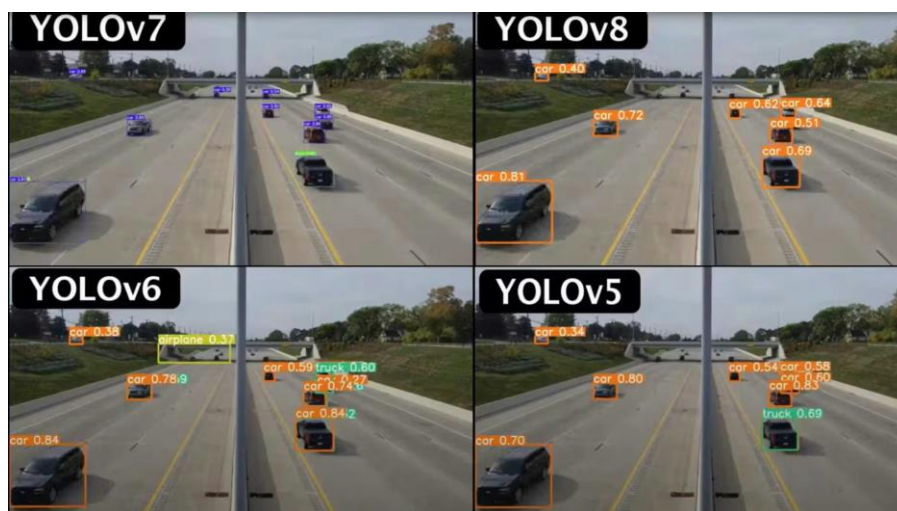


Рисунок 1.6 — Робота різних версій YOLO

YOLOv5, хоч і викликає суперечки через те, що створена сторонніми розробниками (Ultralytics), стала надзвичайно популярною завдяки простоті використання, підтримці PyTorch, широкій документації та активній спільноті. Модель постачається в різних розмірах (YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x), що дозволяє обирати баланс між швидкістю та точністю залежно від потреби. Архітек-

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

тура складається з трьох основних частин: Backbone (екстракція ознак), Neck (агрегація ознак різних масштабів, часто з використанням FPN або PAN), та Head (власне виявлення об'єктів).

Створення власної моделі YOLO — це процес, що охоплює кілька важливих етапів: від підготовки даних і конфігурації архітектури до безпосереднього навчання і тестування. Кожен етап є критично важливим для отримання ефективної, стабільної та точної моделі виявлення об'єктів. У цій статті ми розглянемо, як саме створюється модель YOLO — з технічними деталями та поясненнями внутрішньої логіки роботи.

Перший етап — це збір і підготовка набору даних. Для навчання моделі YOLO необхідно мати зображення та відповідні анотації — інформацію про те, де і які об'єкти розташовані. Анотації мають бути у форматі, який очікує модель. Наприклад, формат YOLO передбачає текстовий файл для кожного зображення, де кожен рядок описує об'єкт у вигляді: `<class_id> <x_center> <y_center> <width> <height>`, де координати нормалізовані відносно ширини та висоти зображення (значення від 0 до 1). Такі анотації можна створити вручну або за допомогою спеціальних інструментів, таких як LabelImg, Roboflow, CVAT тощо.

Після підготовки зображень і анотацій формуються дві головні директорії: одна для зображень (зазвичай `images/`), інша — для міток (`labels/`). Вони, у свою чергу, поділяються на підпапки `train/`, `val/` і, за потреби, `test/`. Далі створюється файл-конфігурація даних (наприклад, `data.yaml`), де вказується шлях до підмножин набору, назви класів і їх кількість.

Другий важливий етап — це визначення архітектури моделі. У сучасних реалізаціях YOLO архітектура моделі задається або через файл YAML (у якому визначено шари, кількість вихідних каналів, розмір фільтрів, з'єднання шарів), або використовується вже готова архітектура із змінною конфігурацією, яка адаптується автоматично під кількість класів. Такі архітектури мають кілька варіантів за розміром: `s`, `m`, `l`, `x` — від найлегшої до найпотужнішої.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

На цьому етапі можна також використати попередньо натреновану модель (pretrained weights). Це дозволяє скоротити час тренування і досягти кращих результатів, оскільки модель уже «бачила» загальні об’єкти та структури на великому наборі. До цього методу вдаються у разі transfer learning, коли модель донавчається на новому специфічному наборі даних.

Після конфігурації архітектури й даних починається етап навчання «training». Він полягає у багаторазовому проходженні («епоках») по навчальному набору. На кожному кроці модель отримує батч зображень, передає їх через мережу та формує передбачення: координати рамок, класи і рівень впевненості. Ці передбачення порівнюються з «правильними» значеннями «ground truth», і на основі цього розраховується функція втрат (loss function).

У YOLO loss-функція складається з трьох основних компонентів. Перший — це bounding box regression loss, яка вимірює, наскільки точно модель передбачила координати рамки (рис.1.7). Зазвичай це варіант функції L1 або L2 (або GIoU/DIoU/CIoU (рис.1.8) для кращої стабільності). Другий компонент — це objectness loss, що відповідає за оцінку ймовірності, що в рамці справді є об’єкт. Третій — classification loss, яка оцінює правильність визначення класу об’єкта. Ці втрати комбінуються у загальну мету оптимізації.

$$loss(x, y) = \begin{cases} 0.5(x - y)^2, & \text{if } |x - y| < 1 \\ |x - y| - 0.5, & \text{otherwise} \end{cases}$$

Рисунок 1.7 — Формула bounding box regression loss

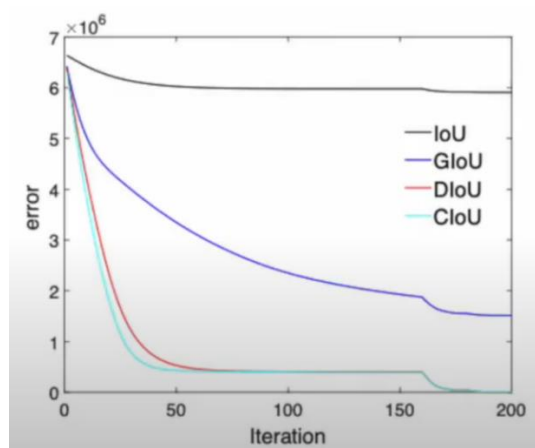


Рисунок 1.8 — Кількість ітерацій та помилок для різних функцій

		Загиривий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

Процес навчання керується оптимізатором — алгоритмом, який змінює ваги мережі для мінімізації втрат. Зазвичай використовується Adam або Stochastic Gradient Descent з моментумом. Також використовуються різні методики стабілізації навчання: learning rate scheduling, warm-up період, регуляризація через DropBlock або Label Smoothing, аугментація зображень (наприклад, Mosaic, Random Flip, HSV Jittering).

Під час тренування, модель періодично оцінюється на валідаційному наборі. Результати зберігаються у вигляді графіків втрат, точності, мап (mean Average Precision). Після завершення навчання зберігається фінальна модель — це набір ваг (weights), який можна використовувати для інференсу (передбачення).

Для подальшого використання модель можна експортувати в різні формати, як-от ONNX, TorchScript, CoreML або TensorRT, що дозволяє запускати її на мобільних пристроях, веб-браузерах або вбудованих системах. Для зручного використання можна створити API або вбудувати модель у систему комп'ютерного зору, яка працює в реальному часі, наприклад, за допомогою OpenCV або FastAPI.

Робота алгоритму починається з попередньої обробки вхідного зображення. Зображення масштабується до фіксованого розміру, наприклад 416x416 або 640x640 пікселів, і нормалізується, тобто значення пікселів переводяться в діапазон від 0 до 1. Це потрібно для стабільної роботи нейронної мережі. Потім зображення перетворюється у тензор, який є багатовимірною матрицею, з якою працює модель.

Перший ключовий компонент YOLO — це так званий backbone. Це глибока згорткова нейронна мережа (CNN), яка відповідає за витяг ознак — особливостей, притаманних різним ділянкам зображення. Backbone проходить по зображенню з використанням фільтрів (ядра згортки), виявляючи низькорівневі патерни (як-от краї та контури), а потім все більш абстрактні ознаки, як форми та текстури. У різних версіях YOLO використовуються різні архітектури для цієї частини: у YOLOv3 це Darknet-53, у YOLOv4 — CSPDarknet53, у YOLOv5 — CSP-базована мережа, а у YOLOv8 — власна Tensor-based архітектура.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

Після того, як ознаки витягнуто, вони передаються на ґонк — проміжну частину архітектури, що займається агрегацією ознак на різних масштабах. Це дозволяє моделі одночасно аналізувати як великі, так і малі об'єкти. Основні архітектурні рішення тут — це FPN (Feature Pyramid Network) та PAN (Path Aggregation Network). FPN передає ознаки з глибших шарів до вищих для збереження високо-рівневої семантики, а PAN забезпечує зворотний шлях і дозволяє деталям з нижчих рівнів підсилювати результат.

Фінальна частина моделі — це голова (head), яка виконує безпосереднє передбачення. Тут кожна точка тензора передбачає набір параметрів: координати обмежувальної рамки (x, y, ширина, висота), рівень впевненості, що рамка містить об'єкт, а також розподіл імовірностей по класах (наприклад, “людина”, “автомобіль”, “велосипед” тощо). Тобто модель повертає тензор розміром $S \times S \times (B \times (5 + C))$, де S — розмір сітки, B — кількість рамок (anchor boxes), 5 — параметри рамки (x, y, w, h, confidence), а C — кількість класів.

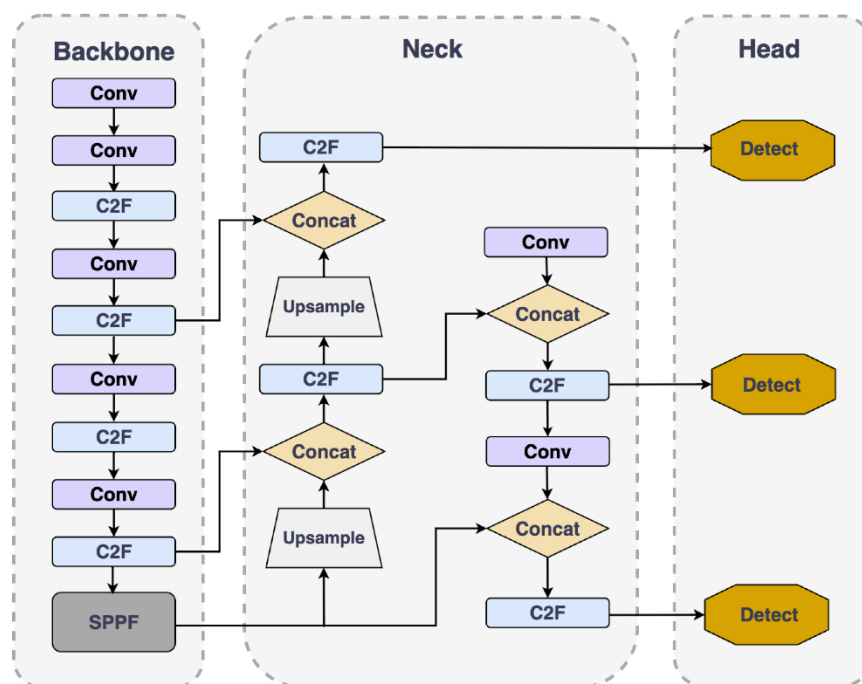


Рисунок 1.9 — Алгоритм розпізнавання об'єктів

YOLO розбиває зображення на сітку (наприклад, 13×13 або 20×20), і кожна клітинка цієї сітки відповідає за передбачення об'єктів, центри яких потрапляють у цю клітинку. Для кожної комірки модель створює кілька обмежувальних рамок (якірних рамок), які оптимізовані для різних співвідношень сторін. Потім на основі

ймовірності наявності об'єкта в кожній рамці та відповідного класу формується фінальний набір детекцій.

Оскільки нейронна мережа може передбачити багато рамок, які перекриваються, після отримання сирих результатів застосовується етап постобробки. Основна техніка тут — Non-Maximum Suppression (NMS), яка залишає лише найбільш релевантну рамку з найвищим рівнем впевненості і відкидає всі інші, які перекриваються з нею на понад певний відсоток (наприклад, 50%).

Процес тренування YOLO включає подачу зображень з анотаціями та мінімізацію втрат, яка включає три основні компоненти: помилку координат (як добре модель передбачила розмір і позицію рамки), втрату класифікації (наскільки правильно визначено клас) і втрату об'єктності (чи є в рамці об'єкт узагалі). Для оптимізації зазвичай використовуються алгоритми типу Stochastic Gradient Descent (SGD) або Adam.

YOLO — це швидкий, потужний та універсальний інструмент комп'ютерного зору. Його постійний розвиток і розширення можливостей роблять його одним із найкращих виборів для реального застосування в завданнях виявлення об'єктів. Незалежно від того, чи ви початківець, чи професіонал у галузі, YOLO — це фундамент, на якому варто будувати проекти, пов'язані з аналізом зображень і відео.

1.5 Огляд бібліотеки комп'ютерного бачення OpenCV

OpenCV — це популярна бібліотека з відкритим кодом для комп'ютерного зору та обробки зображень. Вона була розроблена компанією Intel і вперше випущена у 2000 році. OpenCV активно використовується в галузях штучного інтелекту, робототехніки, розпізнавання облич, медичних систем, безпеки, автоматизації та інших сферах.

OpenCV реалізована на C++, однак має зручні обгортки для таких мов як Python, Java та інші, що дозволяє легко інтегрувати її в найрізноманітніші системи — від вбудованих пристроїв до хмарних сервісів. Головною перевагою бібліотеки є її швидкодія, велика кількість вбудованих алгоритмів, а також активна спільнота та підтримка сучасних технологій штучного інтелекту.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

OpenCV надає широкий набір інструментів(рис.1.10) для обробки зображень і відео: зчитування та збереження медіафайлів, геометричні трансформації, фільтрація, виявлення контурів, сегментація, побудова гістограм, перетворення кольорових просторів, робота з потоковим відео, а також реалізація складних алгоритмів машинного зору — наприклад, виявлення об'єктів, трекінгу руху, розпізнавання облич, розпізнавання жестів, тощо. Окрім цього, бібліотека підтримує інтеграцію з популярними неймережевими фреймворками такими як TensorFlow, PyTorch та ONNX.

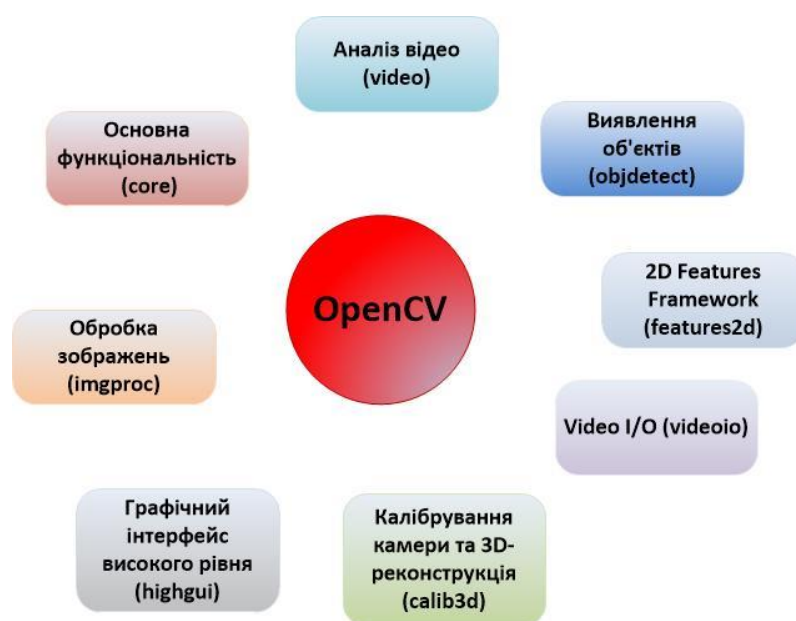


Рисунок 1.10 — Логотип OpenCV

У нашому проєкті, присвяченому розпізнаванню облич на відео в реальному часі, OpenCV відіграє ключову роль на всіх етапах роботи з відеопотоком. Насамперед вона використовується для ініціалізації камери та захоплення кадрів у реальному часі, що є базовим джерелом даних для системи. Далі OpenCV застосовується для обрізання фрагментів зображення, на яких попередньо зафіксовані обличчя за допомогою нейронної мережі YOLO. Вирізані фрагменти передаються до модуля DeerpFace, який відповідає за порівняння облич з базою даних[11].

Ще однією важливою функцією OpenCV у проєкті є візуалізація результатів розпізнавання. З її допомогою на екрані в режимі реального часу відображаються прямокутники навколо виявлених облич, а також підписується ім'я особи, яка була успішно ідентифікована. Завдяки простоті API, ця візуалізація виконується з

		Загрий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

мінімальними затримками, що робить систему зручною для користувача. Окрім того, OpenCV дозволяє зберігати окремі кадри або відеофрагменти, що може бути використано для архівації або подальшого аналізу.

Перевагою використання OpenCV у нашому випадку є її висока продуктивність, простота у використанні та сумісність з іншими інструментами, зокрема з Ultralytics YOLOv8 та DeepFace. Вона дозволяє створити ефективну систему, яка обробляє відео у реальному часі, здійснює детекцію, обрізку, збереження та візуалізацію даних без потреби в додаткових проміжних бібліотеках. Завдяки цьому ми отримуємо компактне, швидке та надійне рішення, яке легко масштабується і підтримує розширення функціоналу в майбутньому[1].

Висновки до розділу

У межах дослідження було розглянуто сучасні методи біометричної ідентифікації особи за голосом, а також відповідні програмні засоби, зокрема: YOLO, OpenCV, DeepFace, FaceNet, TensorFlow та бібліотеку SpeechBrain.

Всі розглянуті технології мають власні переваги та сфери застосування: YOLO виступає ідеальним засобом для детекції обличч в відеопотоці, DeepFace забезпечує високоточну візуальну ідентифікацію, тоді як FaceNet є базовим алгоритмічним компонентом у системах розпізнавання облич. У свою чергу, SpeechBrain дозволяє ефективно вирішувати задачу верифікації особи за голосовим відбитком, поєднуючи передові підходи у сфері обробки мовлення з гнучкістю використання у реальному часі.

З огляду на практичні цілі проєкту — створення надійної системи ідентифікації користувача за допомогою зображення обличчя та голосу — найоптимальнішим рішенням було обрано використання даних технологій. Вони забезпечує високу точність, масштабованість, підтримку попередньо навчених моделей, а також прості в інтеграції з іншими компонентами програмної системи.

Таким чином, вибір було зроблено на користь даних технологій як найбільш ефективного, сучасного та практичного інструменту для реалізації завдань голосової біометричної верифікації у рамках даного проєкту.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

РОЗДІЛ 2 ПРОЄКТУВАННЯ ЗАСТОСУНКУ ДЛЯ ПІДТВЕРДЖЕННЯ ОСОБИСТОСТІ НА ОСНОВІ БІОМЕТРИЧНИХ ХАРАКТЕРИСТИК

2.1 Розробка загального алгоритму функціонування застосунку

Першим кроком роботи буде спроектувати універсальний алгоритм, що реалізує багаторівневу автентифікацію користувача на основі візуальних та голосових біометричних ознак. Архітектура системи сформована за принципами модульності та масштабованості, що забезпечує ефективність, зручність у підтримці, а також можливість подальшого розширення функціоналу. Алгоритм передбачає послідовну взаємодію низки незалежних компонентів, об'єднаних у рамках одного програмного середовища з графічним інтерфейсом користувача.

Проект буде мати таку структуру папок та файлів:

```
| venv
| Security_db
|   |_____| Name1
|   |_____| Photo1.jpg
|   |_____| Photo2.jpg
|   |_____| Photo3.jpg
|   |_____| Voice.wav
|   |_____| Name2
|   |_____| Photo1.jpg
|   |_____| Photo2.jpg
|   |_____| Photo3.jpg
|   |_____| Voice.wav
|   |_____| ...
| Security_db.db
| YOLOv8n-face
| app.py
| gui.py
| database.py
| config.py
| camera.py
| detection.py
| utils.py
| identification.py
| recognition.py
| voice.py
| spectrogram.py
```

venv — це вбудований у Python інструмент для створення ізолюваного віртуального середовища, у якому можна встановлювати пакети незалежно від глобальної системної інсталяції Python. Такий підхід дозволяє уникнути конфліктів між залежностями різних проєктів, особливо коли один і той самий пакет у різних за-

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

стосунках вимагає різних версій. Використання `venv` вважається найкращою практикою при розробці Python-проектів, оскільки воно забезпечує чистоту, передбачуваність і портативність середовища розробки.

В нашому випадку використання `venv` обумовлено використанням Python версії 3.9. Дана версія мови є найстабільнішою для використання `tanserflow` та потрібних йому `dll` файлів.

Структура папок `Security_db` представляє собою дерево папок, логічно розбитих по іменам, де зберігається довільна кількість фотографій персони та еталонний запис її голосу. Кількість фотографій необмежена, чим більше – тим краще. Запис голосу в свою чергу повинен бути тільки в одному екземплярі, довільної довжини, але довжина доріжки більше 5 секунд є непотрібною.

Файл `app.py` є центральною точкою запуску проекту. Ініціалізує створення вікна та асинхронний цикл, за допомогою якого ми в подальшому зможемо запускати функції в асинхронному порядку.

Файл `gui.py` зберігає клас графічного інтерфейсу користувача, створеного за допомогою бібліотеки `PyQt5`, підключає класи розпізнавання облич і голосу, а також координує основну логіку взаємодії між модулями. У ньому формується вікно застосунку, яке включає відеопотік із камери, віджет візуалізації аудіо, та кнопку для запуску голосової перевірки.

Файл `database.py` забезпечує взаємодію з базою даних, де зберігається інформація про користувачів. Таблиця містить поля імені, прізвища, ролі, рівня доступу та шлях до еталонного аудіофайлу. Модуль відповідає за зчитування та перевірку даних користувача під час проходження автентифікації.

Файл `config.py` зберігає всі конфігураційні параметри системи, включаючи шлях до бази облич, вибір моделі розпізнавання, розміри зображення тощо. Він дозволяє централізовано змінювати налаштування системи без потреби втручання в основну логіку кожного модуля.

Файл `voice.py` реалізує процес голосової автентифікації. За допомогою бібліотеки `SpeechBrain`, модуль захоплює мову з мікрофона, проводить її обробку і

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

порівнює з еталонним голосом користувача, збереженим у базі. У разі успішного збігу повертається позитивний результат, який підтверджує справжність особи.

Файл spectrogram.py реалізує графічну візуалізацію звукового сигналу у вигляді осцилограми, розміщеної під вікном камери. Віджет адаптований під дизайн застосунку, не має координатних осей, і змінюється відповідно до активності звуку в реальному часі. Цей компонент підвищує інтерактивність та надає візуальний зворотний зв'язок під час голосової взаємодії.

Подальші файли пов'язані з детекцією та ідентифікацією обличчя в відеопотоці.

Файл camera.py відповідає за захоплення відеопотоку з камери. Він відкриває відповідний пристрій, читає кадри в циклі, і передає їх до модуля детекції. При закритті застосунку або зупиненні зйомки камера коректно звільняється для уникнення ресурсних витоків.

Файл detection.py виконує детекцію облич на відеокадрах у реальному часі. У ньому завантажується модель YOLOv8n-face, після чого вона застосовується до кожного кадру з камери. Результатом роботи цього модуля є координати виявлених облич, які передаються для подальшої ідентифікації.

Файл utils.py містить допоміжні функції, які спрощують обробку зображень, масштабування, відображення підписів або накладання графічних елементів на кадри. Його функції викликаються з інших модулів для покращення читабельності та уникнення дублювання коду.

Файл recognition.py реалізує механізм ідентифікації користувача за зображенням обличчя. У ньому використовується модель DeepFace або її альтернатива для знаходження схожого обличчя у базі даних, сформованій із заздалегідь завантажених зразків. Модуль відповідає за попередню обробку зображення, застосування моделі та повернення імені користувача в разі збігу.

Файл identification.py відповідає за основну логіку розпізнавання облич у реальному часі та взаємодію з камерою в межах окремого потоку. Запускає всі потрібні методи з інших файлів.

		Загрий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

Кожне знайдене обличчя обрізається з зображення, після чого раз у визначений інтервал часу (щоб зменшити навантаження) запускається процедура ідентифікації. Якщо обличчя розпізнане, його ім'я записується в базу даних. Для кожного обличчя на кадрі відображається кольорова рамка і мітка з іменем або "Unknown".

Клас також реалізує сигнал, який передає оброблений кадр назад у графічний інтерфейс, дозволяючи оновлювати вікно з камерою. Таким чином, цей файл виконує роль рушія для розпізнавання облич у реальному часі.

Після запуску додатка активується відеопотік з камери в реальному часі. Кожне виявлене обличчя проходить процедуру попередньої обробки та передається на етап ідентифікації, що реалізований за допомогою DeepFace та дочірньої моделі FaceNet. Модель порівнює вхідне зображення з базою даних облич, що розташована у файловій структурі, і повертає ім'я користувача в разі успішного розпізнавання.

У випадку успішної ідентифікації за обома факторами система виводить на екран інформаційне вікно з персоналізованим привітанням та зазначеним рівнем доступу. У разі невідповідності – виводиться відповідне повідомлення про помилку. Особливістю проєкту є також візуалізація аудіосигналу у вигляді центральної хвилі без осей, що динамічно змінюється залежно від активності мови, гармонійно інтегруючись в інтерфейс нижче відеовікна.

Алгоритм реалізовано з урахуванням вимог до реального часу, стабільності та візуальної ергономіки. Взаємодія всіх компонентів – відеоаналіз, база даних, голосова обробка та графічний інтерфейс – організована асинхронно та оптимізовано з погляду обчислювальних ресурсів. Це дозволяє забезпечити плавну роботу програми на середньостатистичному комп'ютерному обладнанні, що критично важливо для практичного застосування системи в умовах обмежених ресурсів.

Таким чином, загальний алгоритм програми базується на багатоступеневій перевірці користувача, поєднуючи переваги комп'ютерного зору, голосової біометрії та структурованої бази даних у межах цілісної, інтерактивної системи автентифікації.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

2.2 Проєктування бази даних застосунку

Проєктування бази даних є ключовим етапом при створенні програмного за-
собу, що здійснює автентифікацію користувачів за біометричними ознаками. Для
забезпечення збереження та обробки персональних і біометричних даних у системі
необхідно реалізувати централізовану реляційну базу даних на основі СКБД
SQLite. Цей варіант обрано з огляду на простоту інтеграції, мінімальні вимоги до
ресурсів та можливість використання бази даних локально, без окремого сервер-
ного компонента.

У базі даних необхідно передбачити таблицю users, яка міститиме основні
поля, що забезпечують зберігання інформації, необхідної для автентифікації та ав-
торизації користувачів:

- name – текстове поле, яке зберігає ім'я користувача;
- surname – текстове поле для збереження прізвища користувача;
- voice – текстове поле, яке містить шлях до зразка голосу користувача;
- role – поле, що визначає роль користувача в системі (наприклад, “admin”,
“user”, “operator” тощо);
- access – поле, що вказує на рівень доступу або поточний статус авторизації
користувача.

Усі поля(рис.2.1) повинні мати тип TEXT, за винятком поля access, яке може
бути реалізоване як числове, якщо планується градація рівнів доступу.

user	
id (PK)	INTEGER
name	TEXT
surname	TEXT
voice	TEXT
role	TEXT
access	INTEGER

Рисунок 2.1 — Таблиця user

Кожен користувач системи однозначно ідентифікується за іменем, що є клю-
човим полем для пошуку інформації у базі. У межах прикладної логіки система
виконує параметризовані SQL-запити, які гарантують безпечне отримання таких

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

відомостей, як прізвище користувача, його повне ім'я, шлях до голосового шаблону, роль та доступ. Таким чином, забезпечується швидкий і надійний доступ до персоніфікованих даних для наступної обробки в контексті автентифікації.

Окрему увагу слід приділити збереженню контексту роботи користувача. У програмному коді використовується глобальна змінна `tmp_name`, яка тимчасово зберігає ім'я поточного користувача. Це дозволяє уникнути дублювання аргументів у функціях та забезпечує цілісність сесії при зверненні до бази даних. Хоча таке рішення є зручним для прототипу, у майбутньому доцільно реалізувати більш надійну систему керування сесіями, яка буде відповідати вимогам масштабованості та безпеки.

Функціональність, яка передбачається при роботі з базою, охоплює широкий спектр запитів: від простої перевірки імені користувача до отримання біометричних ознак, необхідних для ідентифікації особи. Завдяки уніфікованій структурі таблиці `users`, реалізація таких запитів є ефективною й гнучкою. Усі запити виконуються у форматі 1:1 (один запит — один результат), що відповідає режиму автентифікації за заздалегідь визначеним ідентифікатором.

Загалом, проєктована база даних має забезпечити надійне зберігання біометричних і службових характеристик користувачів, простоту взаємодії з прикладною частиною системи та можливість масштабування структури у разі розширення функціональності. Надалі можливе впровадження додаткових таблиць для збереження історії доступу, логів аутентифікації або взаємодії з додатковими біометричними параметрами.

2.3 Розробка алгоритму ідентифікації обличчя

У межах проєкту передбачається створення алгоритму ідентифікації обличчя, який дозволить встановлювати особу користувача шляхом порівняння його зображення з раніше збереженими біометричними зразками. Основною метою є реалізація надійної системи розпізнавання, яка працює в режимі реального часу, забезпечуючи високу точність і прийнятну швидкість обробки.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

Розробка починається з того, що ми розбиваємо задачу на підзадачі. Отож алгоритм роботи ідентифікатора складається з:

- Запуск вебкамери та зчитування відеопотоку
- Детекція обличчя в відеопотоці
- Обрізання обличчя
- Порівняння з базою даних та рекогнація обличчя
- Обведення обличчя рамкою та вивід результату

В усіх файлах використовуються оператори друку та виклику винятків для більш детального тестування програми.

Для початку необхідно організувати захоплення зображення з відеопотоку вебкамери або іншого підключеного пристрою. Система повинна безперервно отримувати відеокадри та передавати їх на попередню обробку. На цьому етапі планується інтегрувати модель детекції облич, яка визначатиме положення облич у кадрі. Детектор має працювати швидко й ефективно, знаходячи всі обличчя незалежно від їх положення, освітлення чи кута нахилу.

Після виявлення облич на зображенні кожне з них виділяється у вигляді окремого фрагмента для подальшого аналізу. Ці фрагменти повинні бути нормалізовані: приведені до однакового розміру, перетворені у формат, який сумісний із моделлю розпізнавання, та очищені від можливих спотворень.

На етапі розпізнавання планується використання алгоритмів глибокого навчання, зокрема нейромережових моделей, які дозволяють знаходити найбільш схожі обличчя у базі даних. Система має здійснювати порівняння нового зразка з шаблонами, що зберігаються у локальному сховищі. У випадку, коли ступінь схожості з одним із шаблонів перевищує наперед заданий поріг, система вважає особу розпізнаною.

Для забезпечення ефективної роботи алгоритму база даних повинна бути попередньо сформована шляхом реєстрації користувачів. Для кожної особи необхідно зберегти кілька прикладів зображення її обличчя у стандартному форматі. Це дозволить підвищити точність і надійність розпізнавання, особливо в умовах змін зовнішності чи освітлення.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

Також алгоритм має включати механізм обмеження частоти повторного розпізнавання тієї самої особи в короткий проміжок часу. Це дозволить знизити навантаження на систему та уникнути дублювання результатів.

У разі успішного розпізнавання ім'я користувача має бути збережене у тимчасовій змінній, що дозволить надалі використовувати його для отримання повної інформації з бази даних: прізвища, ролі, прав доступу чи інших параметрів.

Для підвищення зручності взаємодії з системою обличчя користувача виводиться на екран разом із іменем, накладеним на відеопотік. Якщо особа не була впізнана, система маркує зону обличчя відповідним повідомленням.

Згідно з вищеописаним алгоритмом роботи створено блок-схему(рис.2.2), яка показує цикл програми та умови, при яких він зупиняється.

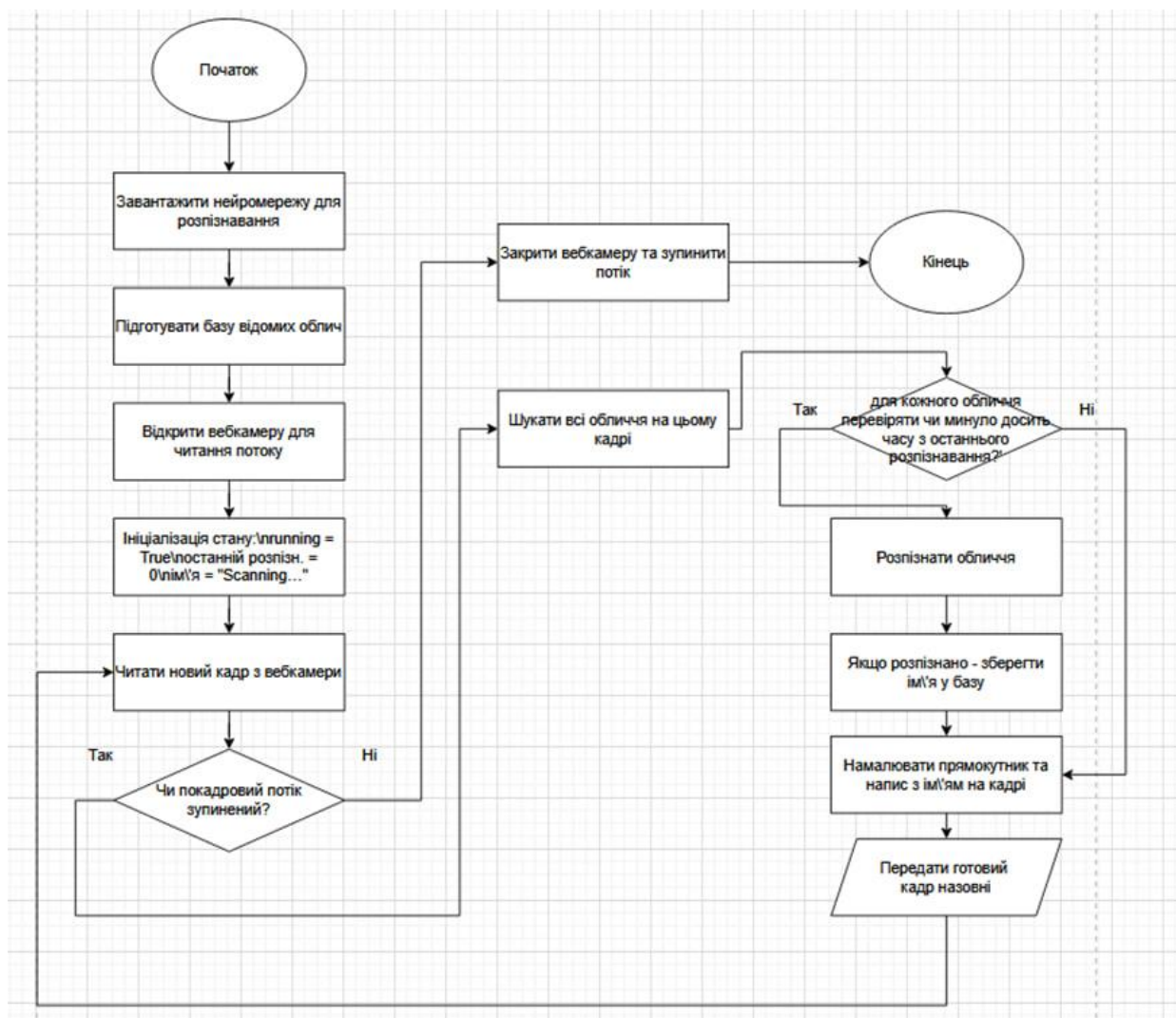


Рисунок 2.2 — Блок-схема алгоритму ідентифікації обличч

Загалом розроблений алгоритм повинен відповідати вимогам стабільної роботи, чутливості до змін обличчя, стійкості до часткових спотворень і бути придатним для роботи в інтерактивному режимі в межах біометричної системи ідентифікації.

2.4 Розробка алгоритму голосової аутентифікації

Ідентифікація особистості за допомогою голосу є потужним біометричним методом, який дозволяє підвищити безпеку у різноманітних інформаційних системах. Ідентифікатор реалізує повноцінну систему перевірки користувача за його голосовими характеристиками, використовуючи бібліотеку SpeechBrain, аудіозапис через sounddevice, та асинхронну обробку з asuncіо.

З метою підвищення надійності біометричної автентифікації користувача, в системі необхідно реалізувати механізм ідентифікації за голосовими ознаками. Голос, як одна з поведенкових біометричних характеристик, володіє достатньою унікальністю і здатністю до відтворення в природних умовах, що робить його зручним і інтуїтивно зрозумілим способом підтвердження особистості.

Розроблений алгоритм голосової аутентифікації має включати декілька послідовних етапів. Першим із них є запис голосового зразка користувача в реальному часі. Для цього система повинна активувати мікрофон пристрою та здійснити запис звукового фрагмента заданої тривалості. Отриманий аудіосигнал підлягає попередній обробці: приведенню до стандартної частоти дискретизації та формату зберігання.

Другий етап полягає у порівнянні записаного зразка з еталонним зразком, що зберігається у базі даних для відповідного користувача. Порівняння виконується за допомогою попередньо навченої моделі розпізнавання мовців, що побудована на нейромережевій архітектурі. Така модель здатна обчислити ступінь схожості між двома голосовими сигналами та видати результат у вигляді числового показника достовірності.

		Загрий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

Оцінювання виконується за допомогою методу верифікації: система отримує два аудіофайли та визначає, чи належать вони одній і тій самій особі. Якщо показник подібності перевищує заздалегідь встановлений поріг, особу вважають автентифікованою.

Згідно з вищеописаним алгоритмом роботи створено блок-схему(рис.2.2), яка показує цикл програми та умови, при яких він зупиняється.

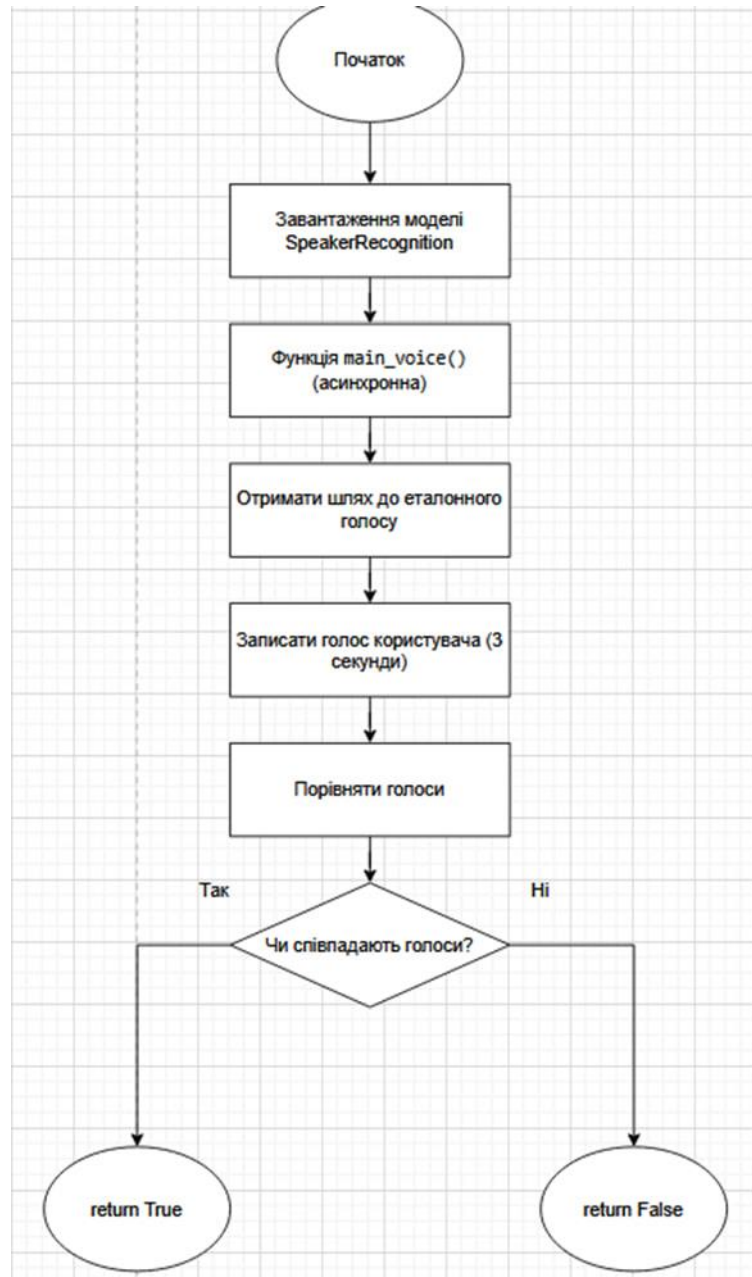


Рисунок 2.3 — Блок-схема алгоритму ідентифікації голосу

Для уникнення затримок у графічному інтерфейсі процес порівняння відбувається в окремому фоновому потоці. Такий підхід дозволяє не блокувати основну програму під час обробки, забезпечуючи одночасну взаємодію з користувачем.

У випадку, якщо в базі не знайдено еталонного голосового зразка, система повинна вивести відповідне повідомлення з попередженням. Це дозволяє уникнути хибних результатів та сигналізує про необхідність реєстрації або повторного збереження голосу.

Для забезпечення стабільності тестування та відлагодження алгоритму, до системи також можна додати можливість автоматичної генерації тестових фраз, які користувач буде вимовляти під час запису. Це забезпечує контроль за змістом мовлення та дозволяє стандартизувати процедуру порівняння.

Також, враховуючи те, що система перевіряє не ключове слово або пароль, а саме голос. Тобто функція дивиться саме на тембр, висоту та звучання голосу. Тому і виникає потреба в додатковій функції, яка просто повертає текст, який буде виведений на екран, для того, щоб персону, яка хоче увійти, читала його[2]. В цей час відбувається запис голосу користувача з метою порівняння. Ця функція повертає випадкове значення з збережених в списку речень.

У результаті реалізації даного алгоритму користувачі отримують зручний та швидкий спосіб автентифікації, який може використовуватись як самостійно, так і в складі багатфакторного підходу до підтвердження особи. Голосова автентифікація значно підвищує рівень безпеки системи та знижує ймовірність доступу сторонніх осіб.

2.5 Проєктування інтерфейсу застосунку

Для забезпечення зручної взаємодії користувача з біометричною системою автентифікації розробляється графічний інтерфейс користувача (GUI), який дозволяє відображати потік із камери, управляти процесом голосової перевірки та наочно виводити результати ідентифікації.

Основним вікном програми виступає центральна панель, де відображається відео з вебкамери. Відеопотік транслюється у реальному часі, а розпізнані обличчя автоматично виділяються відповідними візуальними мітками, що вказують ідентифіковане ім'я або повідомляють про невпізнану особу. Розміри відеоадаптивно

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

підлаштовуються під розміри вікна, що забезпечує стабільне візуальне відображення без спотворень.

У верхній частині інтерфейсу потрібно реалізувати панель керування з назвою застосунку та кнопкою, що ініціює процес голосової автентифікації. При натисканні цієї кнопки система генерує текстову фразу, яку користувач має повторити вголос, після чого запускається перевірка на відповідність голосу з наявними еталонними зразками.

Результати голосової автентифікації виводитимуться у вигляді динамічного повідомлення на напівпрозорій текстовій панелі поверх основного інтерфейсу. Повідомлення можуть містити підтвердження успішної автентифікації з зазначенням імені та ролі користувача, повідомлення про невідповідність голосу або сповіщення про технічну помилку під час перевірки.

У нижній частині головного вікна додатково розміщується візуальний блок для відображення аудіоспектрограми, що ілюструє активність мікрофона під час голосової перевірки. Цей елемент підвищує інформативність інтерфейсу та створює візуальний зв'язок між мовленням користувача та роботою системи.

Для забезпечення цілісного візуального стилю застосунку передбачається єдине оформлення всіх компонентів у темній кольоровій гамі з акцентами на важливі елементи керування. Шрифти, кольори та розміщення елементів відповідають принципам сучасного UX/UI-дизайну, що забезпечує привабливий вигляд і зручність у користуванні(рис.2.4).

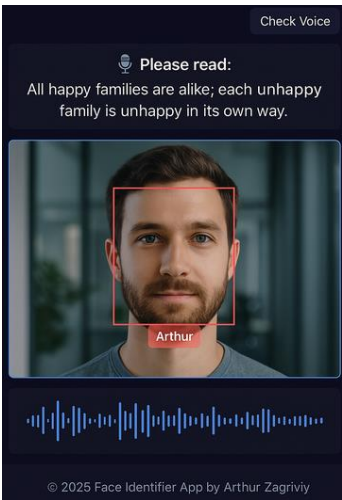


Рисунок 2.4 — Передбачений дизайн інтерфейсу застосунку

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

Також передбачено обробку подій завершення роботи застосунку. Після закриття вікна виконуються процедури звільнення ресурсів, зокрема зупинка роботи модуля відеорозпізнавання та вивільнення камери, що запобігає можливим конфліктам з іншими застосунками.

Загалом, спроектований інтерфейс користувача є інтуїтивно зрозумілим, візуально привабливим та функціонально повним для інтерактивної роботи з системою біометричної ідентифікації. Він поєднує елементи візуалізації, керування та зворотного зв'язку, необхідні для ефективної взаємодії з користувачем у реальному часі.

Висновки до розділу

У межах цього розділу було здійснено всебічне проектування алгоритмів біометричної автентифікації, які забезпечують надійне розпізнавання користувача за обличчям та голосом. Сформульовані рішення охоплюють повний цикл обробки даних — від зчитування біометричного зразка до прийняття рішення про допуск.

Алгоритм ідентифікації за обличчям базується на попередньому виявленні облич у кадрі відеопотоку, їх обрізанні та порівнянні з базою еталонних зразків за допомогою нейромережної моделі. Передбачено механізм періодичного оновлення результатів, що запобігає надмірному навантаженню на систему й забезпечує високу точність розпізнавання в реальному часі.

Голосова автентифікація реалізована шляхом генерації унікальної фрази, яку користувач має вимовити, та порівняння її аудіозапису з раніше збереженим зразком. Для цього залучено спеціалізовану модель верифікації мовців, здатну працювати з короткими аудіофрагментами та визначати ймовірність збігу.

Усі алгоритмічні рішення розроблялися з урахуванням особливостей інтеграції в єдину систему безпеки з базою даних, графічним інтерфейсом користувача та багатопотоковою обробкою. Особливу увагу було приділено гнучкості масштабування, повторному використанню компонентів, а також стійкості системи до помилок розпізнавання. Результати розробки формують надійне підґрунтя для створення цілісної біометричної системи автентифікації, яка поєднує високий рівень безпеки, зручність використання та технологічну адаптивність.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОТОТИПУ ЗАСТОСУНКУ

3.1. Програмна реалізація прототипу застосунку

Вся робота з базою даних відбувається в модулі `database.py`. Всі запити захищені від SQL-ін'єкцій за допомогою параметризованих запитів (?) [17].

Почнемо з створення (рис.3.1) самої бази даних для зберігання інформації про користувачів, що містить таблицю `users` з полями: `name`, `surname`, `voice`, `role`, `access`. Таблиця буде використана іншими модулями через бібліотеку `SQLite` та `SELECT` запитів.

```
conn = sqlite3.connect('security_database.db')

cursor = conn.cursor()

cursor.execute('''
CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT,
    surname TEXT,
    role TEXT,
    access INTEGER,
    voice TEXT
)
''')
```

Рисунок 3.1 — Створення бази даних та таблиці

Після створення бази даних, необхідно попередньо створити користувачів та наповнити їми базу даних (рис.3.2). У кожного користувача буде посилання в таблиці на його еталонний запис голосу.

```
cursor.execute("INSERT INTO users (name, surname, role, access, voice) VALUES ('Arthur', 'Zagriviy', 'administrator', 15, 'security_db/Arthur/Arthur.wav')")
cursor.execute("INSERT INTO users (name, surname, role, access, voice) VALUES ('Olya', 'Lushnikova', 'user', 5, 'security_db/Arthur/Olya.wav')")
cursor.execute("INSERT INTO users (name, surname, role, access, voice) VALUES ('Sophie', 'Zagriva', 'user', 5, 'security_db/Arthur/Sophie.wav')")
cursor.execute("INSERT INTO users (name, surname, role, access, voice) VALUES ('Nikita', 'Drosol', 'user', 5, 'security_db/Arthur/Nikita.wav')")
```

Рисунок 3.2 — Наповнення записами бази даних

У модулі використовується глобальна змінна `tmp_name`, що слугує тимчасовим контейнером для збереження імені користувача після успішної біометричної верифікації. Функція `set_name` (рис.3.3) встановлює значення цієї змінної.

		Загривий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

```

tmp_name = ''

def set_name(name):
    global tmp_name
    tmp_name = name

```

Рисунок 3.3 — Зберігання ідентифікованого імені

Функції `get_surname`, `get_name` та `get_voice_path_by_name`(рис.3.4) дозволяють отримати відповідно прізвище, повне ім'я (ім'я + прізвище) та шлях до еталонного голосового файлу користувача. Функція `get_name()` викликає `get_surname()` і поєднує його з ім'ям. Це дозволяє показувати повну інформацію користувача в GUI.

```

def get_surname():
    global tmp_name
    name = tmp_name
    conn = sqlite3.connect('security_database.db')
    cursor = conn.cursor()

    cursor.execute("SELECT surname FROM users WHERE name = ?", (name,))
    result = cursor.fetchone()
    conn.close()

    return result[0] if result else None

def get_name():
    global tmp_name
    name = tmp_name
    surname = get_surname()
    return name + ' ' + surname

def get_voice_path_by_name():
    global tmp_name
    name = tmp_name
    conn = sqlite3.connect('security_database.db')
    cursor = conn.cursor()

    cursor.execute("SELECT voice FROM users WHERE name = ?", (name,))
    result = cursor.fetchone()
    conn.close()

    return result[0] if result else None

```

Рисунок 3.4 — Функції пошуку еталону голосу та виводу імені

Для управління правами користувача у системі реалізовані функції `get_role_by_name` та `get_access_by_name`(рис.3.5), які повертають відповідні поля з бази. Це дозволяє реалізувати рольову модель безпеки, наприклад, надавати доступ до певних функцій лише адміністраторам або обмежити можливості інших користувачів.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

```

def get_role_by_name():
    global tmp_name
    name = tmp_name
    conn = sqlite3.connect('security_database.db')
    cursor = conn.cursor()

    cursor.execute("SELECT role FROM users WHERE name = ?", (name,))
    result = cursor.fetchone()
    conn.close()

    return result[0] if result else None

def get_access_by_name():
    global tmp_name
    name = tmp_name
    conn = sqlite3.connect('security_database.db')
    cursor = conn.cursor()

    cursor.execute("SELECT access FROM users WHERE name = ?", (name,))
    result = cursor.fetchone()
    conn.close()

    return result[0] if result else None

```

Рисунок 3.5 — Функції пошуку ролей та доступу

Реалізацію алгоритму з ідентифікації обличчя почнемо з запуску вебкамери та зчитування відеопотоку. Один із найпоширеніших способів реалізувати це — за допомогою бібліотеки OpenCV, яка забезпечує гнучкі засоби для роботи з відео. У наведеному прикладі (рис.3.6) реалізовано функцію, яка дозволяє ініціалізувати камеру та перевірити успішність її відкриття.

В усіх файлах використовуються оператори друку та виклику винятків для більш детального тестування програми.

```

def open_camera():
    cap = cv2.VideoCapture(CAMERA_INDEX)
    if not cap.isOpened():
        raise RuntimeError(f"[ERROR] Failed to open camera index {CAMERA_INDEX}")
    return cap

```

Рисунок 3.6 — Функція для зчитування відеопотоку

Імпортується змінна CAMERA_INDEX з зовнішнього конфігураційного файлу. Це дозволяє легко змінювати номер підключеної камери без потреби змінювати основний код.

Функція створює об'єкт, передаючи йому індекс камери. Якщо значення CAMERA_INDEX дорівнює 0, це означає, що буде використано першу камеру,

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

підключену до комп'ютера. OpenCV надає змогу зчитувати відеопотік з різних джерел, включно з файлами, потоками або пристроями — тому це рішення є універсальним.

Після створення об'єкта сар важливо перевірити, чи камера була успішно відкрита. Якщо камера недоступна (наприклад, не підключена або використовується іншою програмою), функція викликає виняток, у повідомленні якого зазначено, який саме індекс не вдалося відкрити. Такий підхід дозволяє одразу зреагувати на помилку та уникнути подальших збоїв у виконанні програми.

Наступним кроком розробляємо функцію для детекції обличчя, але для того, щоб вона працювала треба створити ще функцію для завантаження моделі.

Функція (рис.3.7) відповідає за завантаження моделі з вказаного шляху. В нашому випадку це «YOLOv8n-face». Перед завантаженням виконується перевірка: якщо файл моделі не існує, програма викликає помилку, яка містить повідомлення про відсутність моделі. У разі успіху виконується ініціалізація об'єкта YOLO з відповідним шляхом до моделі. Це дозволяє працювати з попередньо навченим детектором, що готовий до використання без необхідності повторного навчання.

```
def load_yolo_model():  
    if not os.path.exists(YOLO_MODEL_PATH):  
        raise FileNotFoundError(f"[ERROR] YOLO model not found: {YOLO_MODEL_PATH}")  
    return YOLO(YOLO_MODEL_PATH)
```

Рисунок 3.7 — Функція для завантаження моделі

Друга функція (рис.3.8) — виконує безпосереднє виявлення облич на зображенні або кадрі. На вхід вона приймає модель YOLO та зображення: кадр відео. Модель застосовується до зображення, а результати виводяться у вигляді першого елемента списку, що містить передбачення. Далі функція перевіряє, чи модель дійсно виявила об'єкти: якщо так — витягує координати обмежувальних рамок у форматі x_min, y_min, x_max, y_max та перетворює їх у масив NumPy для зручнішої подальшої обробки.

		Загивий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

```
def detect_faces(model, frame):
    results = model(frame, verbose=False)[0]
    boxes = results.bboxes.xyxy.cpu().numpy() if results.bboxes else []
    return boxes
```

Рисунок 3.8 — Функція для детекції обличчя

Повернуті координати будуть використані для передачі в інші модулі системи та обрізаємо обличчя за допомогою бібліотеки OpenCV.

Далі нам треба порівнювати обличчя з базою даних, для цього нам знадобиться три функції: підготовки бази даних, обробки зображення та розпізнавання.

Функція підготовки бази даних (рис.3.9) відповідає за перевірку наявності папки з базою облич і перелік осіб у ній. Вона перевіряє, чи існує вказана директорія, та зчитує імена всіх підкаталогів, які представляють окремих людей. У разі відсутності бази або зображень — виводиться відповідна помилка. Крім того, функція підраховує кількість зображень у кожній папці та виводить у консоль звіт. Це корисно для попереднього контролю повноти даних.

```
def prepare_database():
    if not os.path.exists(DB_FOLDER):
        raise FileNotFoundError(f"[ERROR] DB folder not found: {DB_FOLDER}")

    people = [name for name in os.listdir(DB_FOLDER)
               if os.path.isdir(os.path.join(DB_FOLDER, name))]

    if not people:
        raise RuntimeError(f"[ERROR] No people found in DB folder: {DB_FOLDER}")

    print(f"[INFO] Loaded {len(people)} people into face database:")
    for person in people:
        num_imgs = len([
            f for f in os.listdir(os.path.join(DB_FOLDER, person))
            if f.lower().endswith(('.jpg', '.jpeg', '.png'))
        ])
        print(f"    - {person} ({num_imgs} images)")
```

Рисунок 3.9 — Функція підготовки бази даних

Функція обробки зображення (рис.3.10) виконує просту попередню обробку зображення обличчя. Обличчя масштабується до заданого розміру з файлу конфігурації, що відповідає вимогам моделі.

		Загивий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

```
def preprocess_face(face_img):
    face = cv2.resize(face_img, FACE_IMG_SIZE)
    return face
```

Рисунок 3.10 — Функція обробки зображення

Основна логіка розпізнавання реалізована у функції `recognize_face` (рис.3.10). Вона отримує зображення обличчя, виконує його підготовку, і передає результат у метод пошуку. Ця функція шукає найбільш схожі зображення у базі. Використовуються ключові параметри: шлях до бази, обрана технологія детекції облич та назва моделі, в нашому випадку це FaceNet.

```
def recognize_face(face_img):
    try:
        face = preprocess_face(face_img)
        result = DeepFace.find(
            img_path=face,
            db_path=DB_FOLDER,
            enforce_detection=False,
            detector_backend=DETECTOR_BACKEND,
            model_name=DEEPFACE_MODEL
        )
        if not result[0].empty:
            identity_path = result[0].iloc[0]["identity"]
            person_name = os.path.basename(os.path.dirname(identity_path))
            return person_name
    except Exception as e:
        print(f"[WARN] Recognition error: {e}")
    return "Unknown"
```

Рисунок 3.10 — Функція розпізнавання

Результат пошуку перевіряється на наявність збігів. Якщо відповідність знайдена, функція визначає ім'я особи, витягуючи його з шляху до зображення. Якщо ж виникає помилка, вона буде виведена в консоль, а результатом функції стане "Unknown". Такий підхід дозволяє зберегти стабільність роботи навіть при часткових збоях або неякісних зображеннях.

У системах комп'ютерного зору важливо не лише виявляти об'єкти або розпізнавати обличчя, а й наочно відображати результати аналізу. Тому необхідно реалізувати функцію (рис.3.11), яка дозволяє нанести прямокутник із підписом на зображення або відеокадр.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

```
def draw_label(frame, box, label, color):
    x1, y1, x2, y2 = map(int, box)
    cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
    cv2.putText(frame, label, (x1, y1 - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.6, color, 2)
```

Рисунок 3.11 — Функція малювання рамки навколо обличчя

Функція приймає чотири аргументи: зображення або кадр відео, на якому буде малюватися мітка, координати обмежувального прямокутника (у форматі `[x1, y1, x2, y2]`), текст, який буде відображено над прямокутником та колір лінії та тексту у форматі BGR.

Ця функція використовується у реальному часі під час роботи алгоритмів розпізнавання, щоб наочно показати, кого було виявлено.

Всі раніше описані функції були логічно розділені по файлам, які відповідають за кожну складову процесу ідентифікації. І всі вони використовуються в класі `Recognizer`.

В цьому класі є два основних методи: `__init__()` та `run()`.

У методі `__init__()` виконується ініціалізація ключових компонентів:

- завантажується модель YOLO для детекції обличч;
- готується база даних із зображеннями обличч;
- відкривається відеопотік з камери.

Також встановлюються початкові значення для змінних керування, які потрібні для контролю процесу розпізнавання та частоти виконання.

Головна обчислювальна логіка виконується в методі `run()` (рис.3.11), який автоматично запускається при старті потоку.

У циклі постійно зчитуються кадри з відео. Якщо кадр не вдалося зчитати, цикл переривається. Далі на кожному кадрі виконується детекція обличч за допомогою моделі YOLO. Для кожного виявленого обличчя координати прямокутника використовуються для вирізання відповідної області зображення — це зображення обличчя, яке надалі піддається розпізнаванню.

```

def run(self):
    while self.running:
        ret, frame = self.cap.read()
        if not ret:
            break

        boxes = detect_faces(self.model, frame)
        for box in boxes:
            x1, y1, x2, y2 = map(int, box[:4])
            face = frame[y1:y2, x1:x2]

            now = time.time()
            if now - self.last_recog_time > RECOGNITION_INTERVAL:
                self.last_name = recognize_face(face)
                self.last_recog_time = now
                if self.last_name != "Unknown":
                    set_name(self.last_name)

            color = (247, 90, 90) if self.last_name != "Unknown" else (0, 0, 255)
            draw_label(frame, (x1, y1, x2, y2), self.last_name, color)

        self.frame_ready.emit(frame)

```

Рисунок 3.11 — Метод ідентифікації

Розпізнавання виконується не на кожному кадрі, а з певним інтервалом часу. Це дозволяє зменшити навантаження на систему. Якщо минуло достатньо часу, викликається функція розпізнавання і результат (ім'я особи) зберігається в змінну. Якщо особу вдалося ідентифікувати, її ім'я також передається в базу даних через функцію `set_name()`.

Після отримання імені особи, функція для малювання рамки навколо обличчя накладає на зображення прямокутник і підпис. Колір прямокутника змінюється в залежності від того, чи особу розпізнано: фіолетовий для відомих осіб або червоний для невідомих.

Оновлений кадр передається через сигнал до графічного інтерфейсу, де він відображається у відповідному віджеті.

Ідентифікація особистості за допомогою голосу реалізовуємо використовуючи бібліотеку SpeechBrain, аудіозапис через `sounddevice`, та асинхронну обробку з `asyncio`.

У коді використовується попередньо навчена модель ECAPA-TDNN. Ця модель є однією з найточніших серед відкритих рішень для розпізнавання мовців.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

Вона навчена на великому корпусі VoxCeleb і дозволяє порівнювати два голосові зразки, видаючи імовірність того, що вони належать одній людині.

Функція `record_audio_to_file` (рис.3.12) відповідає за запис голосу з мікрофона. Вона використовує бібліотеку `sounddevice` для отримання сигналу з мікрофона, після чого зберігає його у тимчасовий WAV-файл через `torchaudio`. Таке рішення дозволяє записувати короткі зразки голосу (3 секунди за замовчуванням), достатні для розпізнавання.

```
def record_audio_to_file(duration=3, fs=16000):
    recording = sd.rec(int(duration * fs), samplerate=fs, channels=1, dtype='float32')
    sd.wait()

    tmpfile = tempfile.NamedTemporaryFile(suffix=".wav", delete=False)
    waveform = torch.tensor(recording).T
    torchaudio.save(tmpfile.name, waveform, fs)
    return tmpfile.name
```

Рисунок 3.12 — Функція для запису голосу

Функція порівняння голосів (рис.3.13) приймає два аудіофайли: один — з бази даних (еталонний), інший — записаний користувачем. За допомогою моделі розраховується схожість між ними. Результатом є числове значення `score` (від 0 до 1) і булевий показник `prediction`.

```
def compare_voices(file1, file2):
    score, prediction = recognizer.verify_files(file1, file2)
    return score.item(), prediction
```

Рисунок 3.13 — Функція порівняння голосів

Функція `main_voice` (рис.3.14) є асинхронною, що дозволяє запускати голосову перевірку у фоновому режимі. Це досягається за допомогою бібліотеки `асунсіо`, яка є вбудованою в Python. Для того, щоб асинхронність працювала, необхідно створити окремий асинхронний цикл, який буде працювати в іншому потоці. В цьому циклі і будуть запускатися всі інші асинхронні функції та методи.

Це необхідно для коректної роботи застосунку. Так, як функція буде працювати в іншому потоці, вона не буде блокувати роботу основного вікна, створеного на PyQt5. Вона делегує виконання синхронних функцій до окремого потоку через `ThreadPoolExecutor`.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

```

async def main_voice():
    loop = asyncio.get_event_loop()

    def sync_voice_check():
        reference_file = get_voice_path_by_name()
        if not reference_file:
            return "❗ Reference voice not found"

        recorded_file = record_audio_to_file(duration=3)
        score, is_same = compare_voices(reference_file, recorded_file)

        return True if is_same else False

    try:
        result = await loop.run_in_executor(executor, sync_voice_check)
        return result
    except Exception as e:
        return f"❌ Помилка: {e}"

```

Рисунок 3.14 — Асинхронна функція для запуску перевірки

Також, враховуючи те, що система перевіряє не ключове слово або пароль, а саме голос. Тобто функція дивиться саме на тембр, висоту та звучання голосу. Тому і виникає потреба в додатковій функції, яка просто повертає текст, який буде виведений на екран, для того, щоб персону, яка хоче увійти, читала його[2]. В цей час відбувається запис голосу користувача з метою порівняння. Ця функція повертає випадкове значення з збережених в списку речень.

У межах реалізації біометричної системи розпізнавання особистості важливу роль відіграє модуль `gui.py`, який відповідає за побудову графічного інтерфейсу. Завдяки інтеграції з бібліотекою `PyQt5`, інтерфейс забезпечує зручну візуалізацію процесів розпізнавання обличчя та голосу, а також взаємодію користувача з системою.

Модуль `gui.py` реалізує головне вікно програми через клас `MainWindow`, який наслідує `QMainWindow` із `PyQt5`[8]. Інтерфейс побудований відповідно до принципу модульності: логіка взаємодії з відео, голосом і базою даних винесена в окремі файли тоді як сам `gui.py` займається лише побудовою інтерфейсу та керуванням подіями.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

Всі елементи інтерфейсу, такі як заголовок, відеоекран, кнопки та нижній колонтитул, розміщуються за допомогою вертикальних і горизонтальних лейаутів. Це забезпечує адаптивність вікна при зміні його розміру.

Однією з головних функцій інтерфейсу є трансляція відеопотоку з камери в реальному часі. Це реалізується через клас Recognizer, який надсилає кадри до інтерфейсу у вигляді сигналу. Метод `update_frame`(рис.3.15) обробляє кадри, обрізає їх до бажаного формату, конвертує в колірний простір RGB і виводить на екран у віджеті `video_label`. Це забезпечує постійний візуальний контроль за процесом ідентифікації обличчя, а також створює відчуття «живого» інтерфейсу.

```
def update_frame(self, frame):
    cropped = self.crop_center(frame, target_ratio=self.video_label.width() / self.video_label.height())
    rgb_image = cv2.cvtColor(cropped, cv2.COLOR_BGR2RGB)
    h, w, ch = rgb_image.shape
    bytes_per_line = ch * w
    qimg = QImage(rgb_image.data, w, h, bytes_per_line, QImage.Format_RGB888)

    self.video_label.setPixmap(QPixmap.fromImage(qimg).scaled(
        self.video_label.size(), Qt.KeepAspectRatio, Qt.SmoothTransformation))
```

Рисунок 3.15 — Функції пошуку ролей та доступу

Користувач може пройти голосову перевірку, натиснувши кнопку “Check Voice”. Ця подія запускає асинхронний метод `async_check_voice`(рис.3.16), що дозволяє не блокувати головний потік інтерфейсу. Спочатку система генерує фразу, яку потрібно озвучити, а потім проводить звірку з еталонним голосом користувача.

```
async def async_check_voice(self):
    self.show_overlay_text("🔊 Generating sentence...")
    input_text = generate_text()
    self.show_overlay_text("👂 Please read:\n\n" + input_text)

    is_true = await main_voice()

    if is_true is True:
        name = get_name()
        role = get_role_by_name()
        self.show_overlay_text(f"✅ Welcome to the system,\n{name}\nYour role is {role}", duration_ms=3000)
    elif is_true is False:
        self.show_overlay_text("❌ Voice does not match", duration_ms=3000)
    else:
        self.show_overlay_text(f"⚠️ Error:\n{is_true}", duration_ms=4000)
```

Рисунок 3.16 — Функції пошуку ролей та доступу

Результат перевірки відображається у вигляді текстового повідомлення на екрані, яке виводиться за допомогою спеціального елемента `overlay_label`. Це напівпрозорий блок, що з'являється поверх інтерфейсу на кілька секунд і містить зрозумілі повідомлення про успішний вхід, помилку або збій.

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

Система активно використовує overlay-механізм для взаємодії з користувачем. Наприклад, при запуску голосової перевірки з'являється інструкція, яку потрібно прочитати. Після перевірки — чітке повідомлення про результат. Такий підхід значно покращує досвід користувача, дозволяючи уникати неочевидних дій або непорозумінь.

Крім того, в інтерфейсі передбачено місце для аудіоіндикатора — кастомного виджета AudioWaveformWidget, який дозволяє візуалізувати звукову хвилю під час запису голосу. Це ще більше підвищує рівень зручності та наочності.

3.2. Тестування роботи застосунку

Після запуску програми відкривається головне вікно(рис.3.17), в якому відображається інтерфейс користувача з мінімалістичним, але інформативним дизайном. У верхній частині розміщено заголовок та кнопка "Check Voice", що відповідає за запуск голосової перевірки. Центральну частину займає вікно відеопотоку з камери, де відображається зображення користувача в реальному часі. Нижче виводиться аудіовізуальний індикатор(рис.3.18), який демонструє форму аудіосигналу при голосовому введенні.

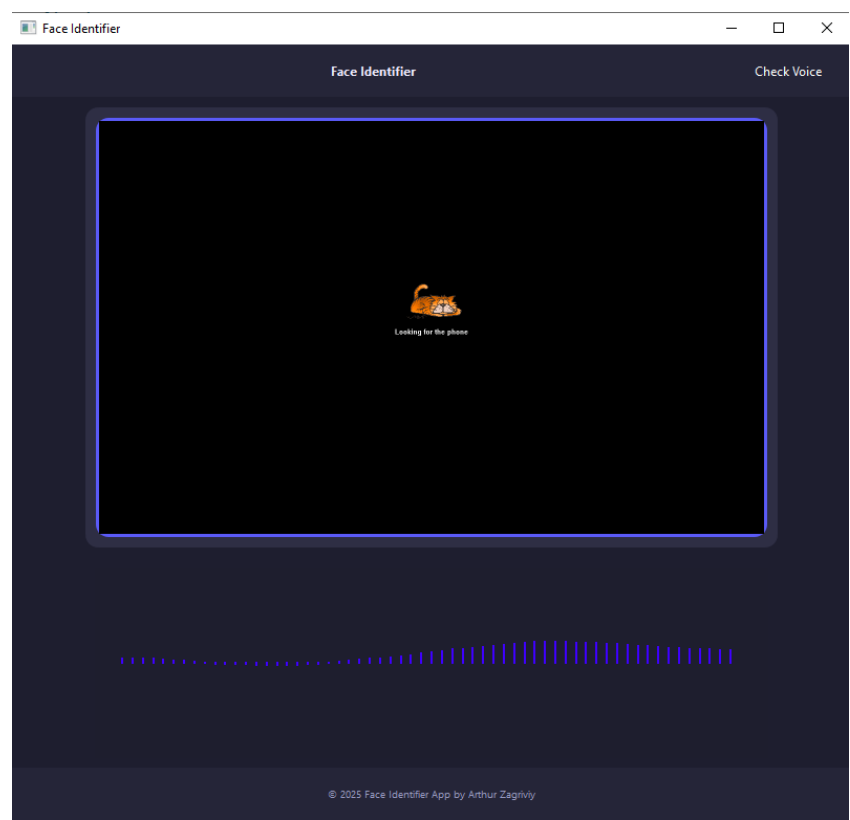


Рисунок 3.17 — Головне вікно

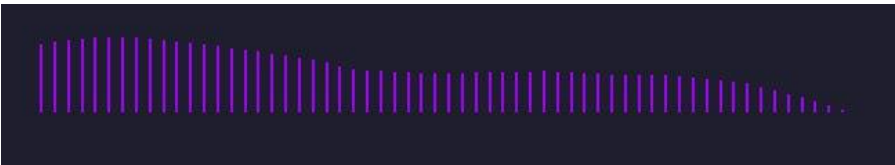


Рисунок 3.18 — Аудіовізуальний індикатор

Інтерфейс є динамічним та адаптивним: він автоматично підлаштовується під розмір вікна користувача, зберігаючи правильні пропорції елементів.

Одразу після запуску та завантаження активується модуль відеорозпізнавання (рис.3.19), який аналізує потік з камери.

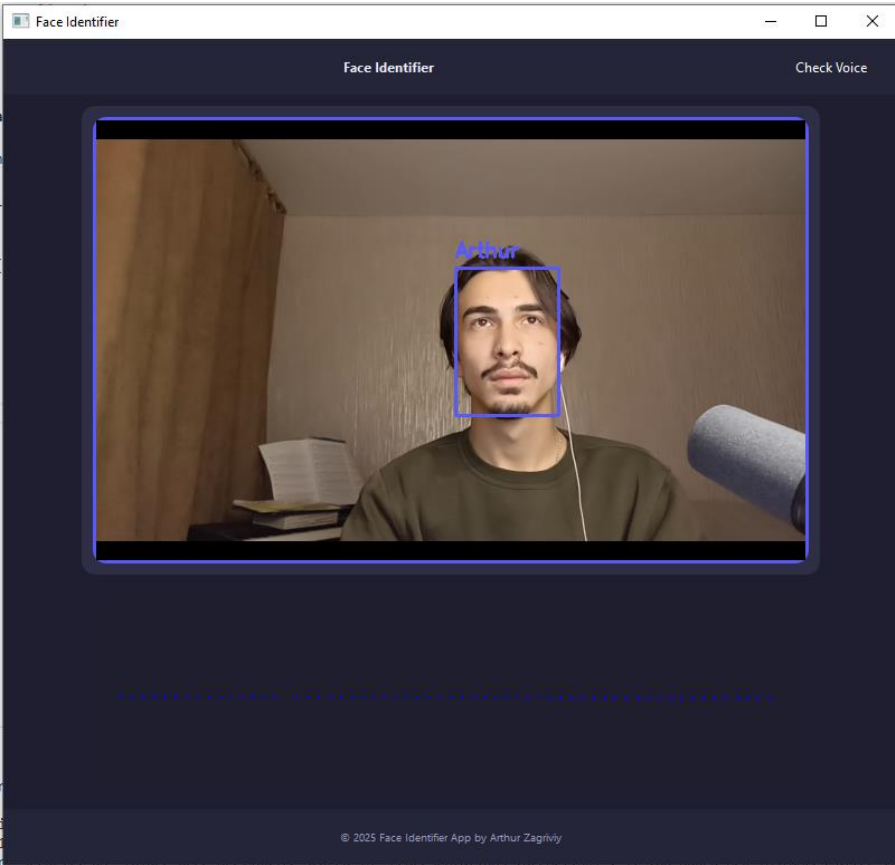


Рисунок 3.19 — Аудіовізуальний індикатор

Варто відзначити, що завдяки механізму попередньої обрізки зображення до цільового співвідношення сторін, забезпечується краща фокусованість на обличчі користувача, що покращує точність розпізнавання навіть при слабкому освітленні або частковому перекритті кадру.

Після натискання кнопки “Check Voice”, система генерує випадкову фразу(рис.3.20), яку користувач повинен чітко прочитати в мікрофон. Генерація фраз дозволяє уникнути шаблонного розпізнавання та ускладнює спроби підробки голосу через відтворення запису.

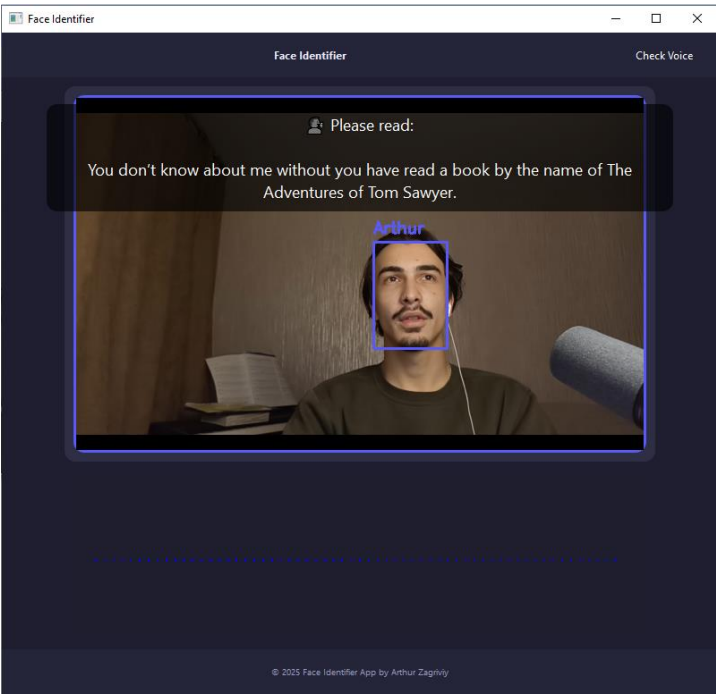


Рисунок 3.20 — Генерація фрази для перевірки голосу

Під час запису голосу відображається аудіосигнал у реальному часі — це не лише зручно для користувача, а й дозволяє переконатися в тому, що мікрофон працює справно.

У разі успішного збігу голосу та обличчя, користувач отримує привітальне повідомлення з зазначенням його ПІБ та ролі (рис.3.21). У разі невдалої верифікації виводиться повідомлення про невідповідність голосу або виникнення помилки.

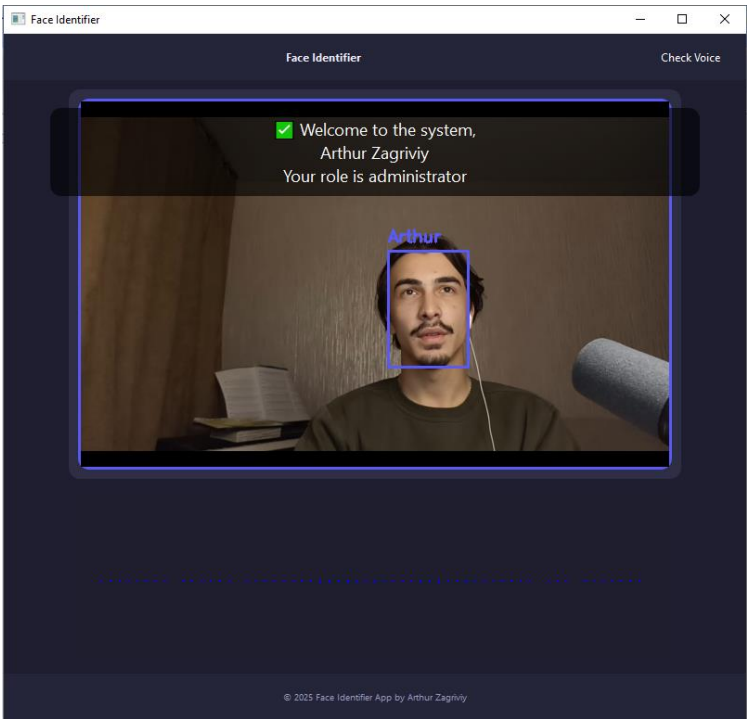


Рисунок 3.21 — Успішна автентифікація

Додаток працює досить стабільно. При поворотах голови ідентифікація працює добре(рис.3.22); в теорії, це і є основна специфіка DeerFace. З поганим світлом також все працює стабільно.

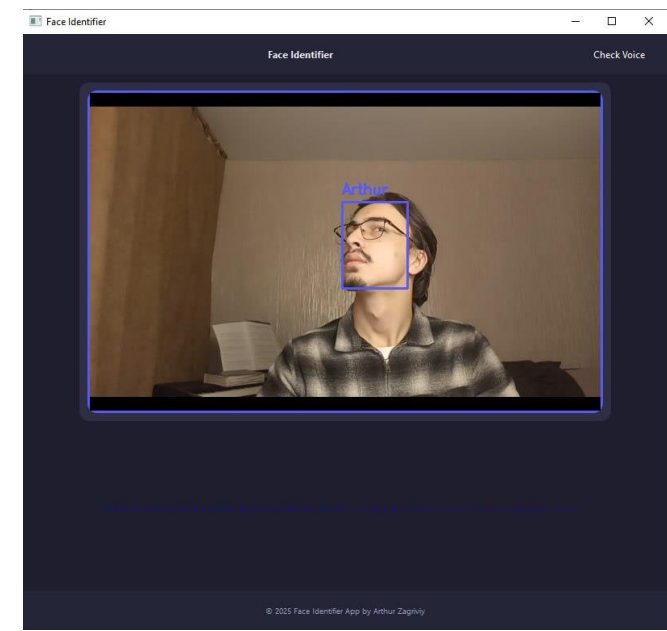


Рисунок 3.22 — Ідентифікація при повороті голови

Також варто зазначити, що ідентифікатор при збігах деяких обставин може некоректно працювати. Наприклад, якщо обличчя закрити рукою, то персону або буде некоректно ідентифіковано (рис.3.23), або неопізнано зовсім (рис.3.24). В разі некоректної ідентифікації, порушник не зможе увійти до системи, через аудіо-ідентифікацію.

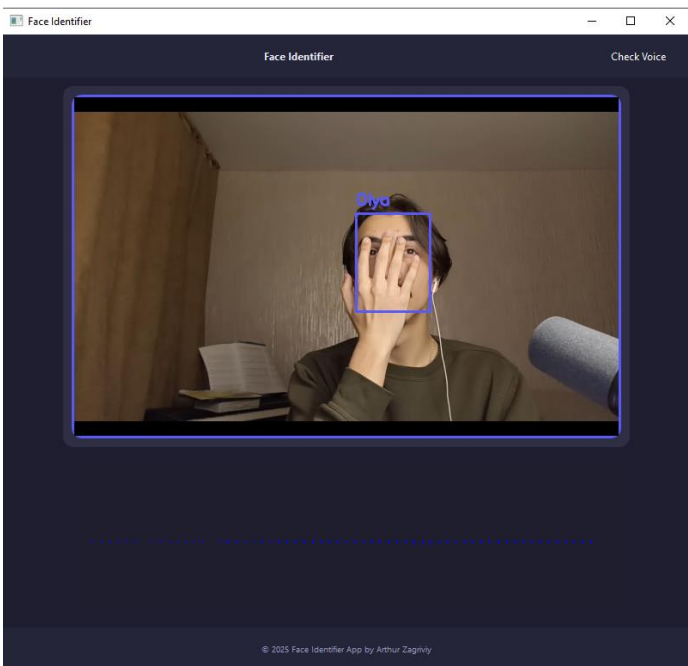


Рисунок 3.23 — Некоректна ідентифікація

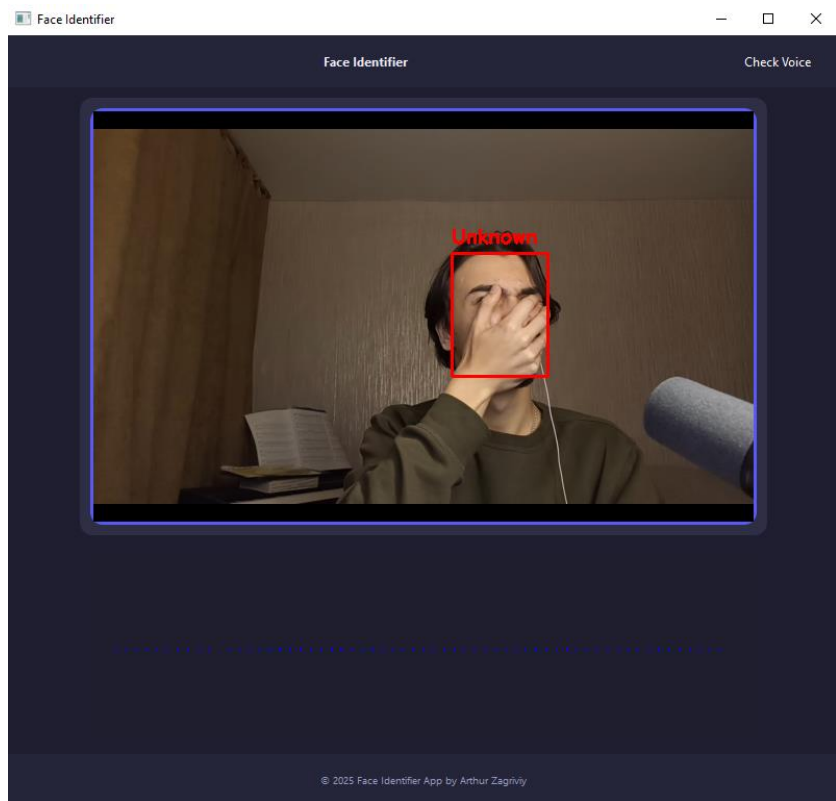


Рисунок 3.24 — Невдала спроба ідентифікації

Аудіо-ідентифікація працює відмінно, навіть при некритичних змінах голосу, змінах тембру. Також програмний код, який реалізовує ідентифікацію через голос, значно менше потребує ресурсів комп'ютера, ніж візуальна ідентифікація.

Висновки до розділу

Ретельна розробка та інтеграція модулів комп'ютерного зору й аудіоаналізу є ключовим фактором у забезпеченні надійності та точності біометричної системи аутентифікації. Завдяки використанню передових технологій, таких як YOLO для виявлення облич, DeepFace для верифікації особи та SpeechBrain для голосової перевірки, система демонструє високу ефективність при ідентифікації користувачів у реальному часі.

Однією з головних переваг є мультифакторний підхід, при якому кожен етап (виявлення, розпізнавання, голосова перевірка) виконує окрему функцію, але в комплексі формує надійний захист від несанкціонованого доступу. Наприклад, візуальна перевірка особи дозволяє попередньо фільтрувати користувачів, тоді як голосова перевірка виключає підробки на основі фотографій чи відео.

		Загивий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

Особливу увагу приділено обробці винятків і помилок — система здатна повідомляти про відсутність бази осіб, недостатню кількість зразків чи помилки при порівнянні, що є важливим для зручного обслуговування й масштабування.

Проте, як і будь-яка складна система, запропоноване рішення має свої обмеження. Чутливість до умов освітлення, шумів мікрофона або варіацій голосу може вплинути на стабільність розпізнавання. Крім того, глибина результатів залежить від якості вхідних даних і попередньої підготовки бази — відсутність достатньої кількості зображень чи записів голосу призводить до зниження точності.

Загалом, поєднання відеоаналізу та голосової біометрії створює гнучку, розширювану основу для побудови сучасної системи контролю доступу. Проте її ефективне використання потребує регулярного оновлення баз даних, коректного налаштування моделей та обережного підходу до обробки біометричної інформації з урахуванням етичних та правових аспектів.

Також проведена демонстрація та тестування розробленої системи біометричної ідентифікації підтвердили її функціональність, зручність у використанні та здатність точно ідентифікувати користувача за комбінацією візуальних і голосових характеристик. Завдяки поєднанню модулів розпізнавання обличчя в реальному часі та перевірки голосу, програмне забезпечення демонструє високий рівень достовірності при автентифікації.

У ході тестування було виявлено, що система успішно розпізнає зареєстрованих користувачів навіть за наявності незначних змін у зовнішності або інтонації голосу. Разом із тим, надійність розпізнавання знижується при співпадінні або наявності деяких факторів, що свідчить про потребу у подальшій оптимізації моделей та розширенні алгоритмів обробки помилок.

Загалом, реалізована система підтвердила свою працездатність у тестовому середовищі. Вона може бути використана як базова модель для створення більш складних систем контролю доступу, що включають багатофакторну аутентифікацію, роботу з мережевими базами даних, логування спроб входу та інтеграцію з фізичними системами безпеки.

		Загрий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

ВИСНОВКИ

У кваліфікаційній роботі було реалізовано повноцінну систему біометричної ідентифікації особи з використанням технологій комп'ютерного зору та розпізнавання голосу. Основною метою дослідження було створення програмного забезпечення, яке забезпечує надійний доступ до інформаційної системи лише авторизованим користувачам, шляхом поєднання двох незалежних факторів біометричної автентифікації — розпізнавання обличчя та перевірки голосу. У процесі виконання роботи було послідовно вирішено низку завдань, що охоплюють аналіз, проектування, розробку, інтеграцію та тестування програмної системи.

На початковому етапі було проведено огляд сучасних підходів до біометричної автентифікації, а також розглянуто ключові поняття та алгоритми, що використовуються у сфері комп'ютерного зору (зокрема, технологія Face Recognition, яка базується на глибокому навчанні) та обробки мовлення (із застосуванням генерації спектрограм та методів голосової верифікації). Аналіз наявних рішень у цій галузі дозволив сформулювати основні вимоги до функціональності, безпеки та зручності використання майбутнього програмного продукту.

У практичній частині роботи було розроблено графічний інтерфейс користувача за допомогою бібліотеки PyQt5, що забезпечує інтерактивну взаємодію з системою та виведення в реальному часі відеопотоку з камери, аудіосигналу та текстових підказок. Було реалізовано модуль розпізнавання обличчя на основі нейронних мереж, який здатен стабільно працювати в режимі реального часу, а також модуль перевірки голосу, що використовує попередньо згенеровані текстові фрази для голосового підтвердження особистості користувача.

Особливу увагу приділено роботі з базою даних, де зберігається інформація про користувачів: імена, прізвища, ролі в системі, рівні доступу, а також посилання на їхні голосові шаблони. Це дозволяє не лише проводити ідентифікацію, але й персоналізувати повідомлення інтерфейсу та враховувати рольову модель доступу в рамках інформаційної безпеки.

У розділі демонстрації та тестування було показано практичну роботу системи в реальному середовищі. Система стабільно ідентифікує користувачів навіть

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

при змінному освітленні, завдяки попередньому кадруванню зображення та центруванню обличчя. Було підтверджено ефективність асинхронної обробки даних при перевірці голосу, що дозволяє не блокувати інтерфейс програми під час верифікації.

У процесі тестування виявлено кілька потенційних обмежень, зокрема зниження точності частковому закритті обличчя, затримках при завантаженні відеопотоку на слабших комп'ютерах або проблемах із мікрофоном. Проте ці обмеження є типовими для систем реального часу і можуть бути зменшені шляхом подальшої оптимізації або розширення апаратного забезпечення.

Загалом, результати роботи підтверджують доцільність поєднання декількох методів біометричної автентифікації для підвищення безпеки систем контролю доступу. Створене програмне забезпечення демонструє не лише працездатність у тестовому середовищі, але й потенціал до масштабування, інтеграції з іншими сервісами або апаратними засобами безпеки, такими як замки, картридери, електронні системи моніторингу тощо.

Варто також відзначити, що гнучка архітектура коду дозволяє розширювати функціональність проєкту без значного переписування існуючих модулів. Зокрема, можуть бути реалізовані додаткові механізми аутентифікації (наприклад, розпізнавання відбитків пальців або райдужної оболонки ока), підтримка хмарного зберігання даних користувачів, ведення журналу входів та спроб доступу.

Підсумовуючи, можна зазначити, що в межах дипломної роботи було досягнуто поставленої мети — реалізовано функціональну, адаптивну та ефективну систему біометричної ідентифікації користувачів. Отримані результати мають не лише навчально-практичне, але й прикладне значення, адже впровадження подібних рішень у різні сфери (освітні установи, офіси, приватні домогосподарства) сприятиме підвищенню рівня безпеки та контролю над доступом до інформаційних і фізичних ресурсів.

		Загивий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sanderson C., Paliwal K. Face recognition using speech and visual cues: Review and recommendations // Pattern Recognition. – 2003. – №36(9). – С. 2455–2467. (розділ "Огляд біометричних методів")
2. Kinnunen T., Li H. An overview of text-independent speaker recognition: From features to supervectors // Speech Communication. – 2010. – №52(1). – С. 12–40. (розділ "Теоретичні основи голосової ідентифікації")
3. He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition // Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). – 2016. – С. 770–778. (розділ "Нейронні мережі для розпізнавання обличчя")
4. Parkhi O. M., Vedaldi A., Zisserman A. Deep Face Recognition // British Machine Vision Conference. – 2015. – С. 41.1–41.12.
5. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet classification with deep convolutional neural networks // Communications of the ACM. – 2017. – №60(6). – С. 84–90.
6. PyTorch Documentation. (Електронний ресурс). – Режим доступу: <https://pytorch.org/docs/stable/index.html>
7. Face Recognition Python Library. (Електронний ресурс). – Режим доступу: https://github.com/ageitgey/face_recognition
8. PyQt5 Documentation. (Електронний ресурс). – Режим доступу: <https://doc.qt.io/qtforpython/>
9. Python Speech Recognition Library. (Електронний ресурс). – Режим доступу: <https://pypi.org/project/SpeechRecognition/>
10. NumPy Documentation. (Електронний ресурс). – Режим доступу: <https://numpy.org/doc/>
11. OpenCV Documentation. (Електронний ресурс). – Режим доступу: <https://docs.opencv.org/>
12. Zhang C., Koishida K. End-to-End Text-Independent Speaker Verification with Triplet Loss on Short Utterances // INTERSPEECH. – 2017. – С. 1487–1491.

		Загрийвий А.С.			<i>ККІК.КР.Б – 125 – 08 – ПЗ</i>	Арк.
		Єфіменко А.А.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Шевченко Д. І., Коваленко І. Б. Системи розпізнавання мовлення: огляд та напрямки розвитку // Наукові вісті. – 2020. – №1(127). – С. 18–24.
14. Ivanov V., Lebedev V. Application of CNN in real-time face detection system // Computer Science Journal. – 2021. – №2(4). – С. 45–52.
15. Understanding MFCC Features for Speech Recognition. (Електронний ресурс). – Режим доступу: <https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html>
16. Database SQLite: Офіційна документація. (Електронний ресурс). – Режим доступу: <https://www.sqlite.org/docs.html>
17. Google Speech API Documentation. (Електронний ресурс). – Режим доступу: <https://cloud.google.com/speech-to-text/docs>
18. Zhang, H., Xu, C., Wang, Y. Improving Face Recognition Performance under Real-World Conditions: Challenges and Solutions // Journal of Visual Communication. – 2019. – №5(2). – С. 35–47.

		Загивий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

ДОДАТКИ

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг app.py:

```
import sys
from PyQt5.QtWidgets import QApplication
import gui
import asyncio

if __name__ == "__main__":
    app = QApplication(sys.argv)

    from qasync import QEventLoop
    loop = QEventLoop(app)
    asyncio.set_event_loop(loop)

    window = gui.MainWindow()
    window.show()

    with loop:
        loop.run_forever()

    sys.exit(app.exec_())
```

Лістинг gui.py:

```
from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QWidget, QLabel, QVBoxLayout,
    QFrame, QSizePolicy, QPushButton, QHBoxLayout
)
from PyQt5.QtGui import QImage, QPixmap, QFont
from PyQt5.QtCore import Qt, QTimer
import cv2
import asyncio

from identification import Recognizer
from spectrogram import AudioWaveformWidget
from voice import main_voice, generate_text
from database import get_name, get_role_by_name

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Face Identifier")
        self.resize(800, 600)

        self.setStyleSheet("""
            QWidget {
                background-color: #1E1E2F;
                color: #EEEEAFD;
                font-family: 'Segoe UI';
                font-size: 12px;
            }
            QPushButton {
                background-color: #5A5AF7;
                border: none;
                border-radius: 8px;
                padding: 4px 10px;
            }
        """)
```

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        color: white;
    }
    QPushButton:hover {
        background-color: #7373f7;
    }
    """

self.setup_ui()

self.recognizer = Recognizer()
self.recognizer.frame_ready.connect(self.update_frame)
self.recognizer.start()

# === Прозорий оверлей для тексту ===
self.overlay_label = QLabel("", self)
self.overlay_label.setAlignment(Qt.AlignCenter)
self.overlay_label.setStyleSheet("""
    background-color: rgba(0, 0, 0, 160);
    color: white;
    font-size: 18px;
    border-radius: 12px;
    padding: 10px;
    """)
self.overlay_label.setWordWrap(True)
self.overlay_label.hide()

def show_overlay_text(self, text: str, duration_ms: int = 0):
    self.overlay_label.setText(text)
    self.overlay_label.adjustSize()

    self.overlay_label.resize(self.width() - 100, self.overlay_label.height())
    self.overlay_label.move(
        (self.width() - self.overlay_label.width()) // 2,
        80
    )
    self.overlay_label.show()

    if duration_ms > 0:
        QTimer.singleShot(duration_ms, self.overlay_label.hide)

def check_voice(self):
    asyncio.create_task(self.async_check_voice())

async def async_check_voice(self):
    self.show_overlay_text("🔊 Generating sentence...")
    input_text = generate_text()
    self.show_overlay_text("🗣️ Please read:\n\n" + input_text)

    is_true = await main_voice()

    if is_true is True:
        name = get_name()
        role = get_role_by_name()
        self.show_overlay_text(f"✅ Welcome to the system, \n{name} \nYour role is {role}", duration_ms=3000)
    elif is_true is False:

```

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

```

        self.show_overlay_text("✗ Voice does not match", duration_ms=3000)
    else:
        self.show_overlay_text(f"⚠ Error:\n{is_true}", duration_ms=4000)

def setup_ui(self):
    main_widget = QWidget()
    main_layout = QVBoxLayout(main_widget)
    main_layout.setContentsMargins(0, 0, 0, 0)
    main_layout.setSpacing(0)

    header = QWidget()
    header.setFixedHeight(50)
    header.setStyleSheet("background-color: #252538;")
    header_layout = QHBoxLayout(header)
    header_layout.setContentsMargins(10, 0, 10, 0)
    header_layout.setSpacing(10)

    title = QLabel("Face Identifier")
    title.setFont(QFont("Segoe UI", 18, QFont.Bold))
    title.setAlignment(Qt.AlignCenter)
    title.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Preferred)
    header_layout.addWidget(title)

    self.side_button = QPushButton("Check Voice")
    self.side_button.setFixedSize(100, 30)
    self.side_button.clicked.connect(self.check_voice)
    header_layout.addWidget(self.side_button)

    camera_frame = QFrame()
    camera_frame.setStyleSheet("""
        background-color: #2E2E44;
        border-radius: 12px;
    """)
    camera_layout = QVBoxLayout(camera_frame)
    camera_layout.setContentsMargins(10, 10, 10, 10)

    self.video_label = QLabel("Starting camera...")
    self.video_label.setStyleSheet("""
        border: 3px solid #5A5AF7;
        background-color: #101018;
    """)
    self.video_label.setAlignment(Qt.AlignCenter)
    self.video_label.setFixedSize(640, 400)
    camera_layout.addWidget(self.video_label)

    center_layout = QVBoxLayout()
    center_layout.setAlignment(Qt.AlignCenter)
    center_layout.addWidget(camera_frame)
    center_layout.setContentsMargins(0, 10, 0, 10)
    center_layout.setSpacing(10)

    self.audio_widget = AudioWaveformWidget()

    footer = QLabel("© 2025 Face Identifier App by Arthur Zagriviy")
    footer.setFixedHeight(50)
    footer.setAlignment(Qt.AlignCenter)
    footer.setStyleSheet("background-color: #252538; color: #999AC1; font-size: 10px;")

```

		Загравий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

main_layout.addWidget(header)
main_layout.addLayout(center_layout)
main_layout.addWidget(self.audio_widget)
main_layout.addWidget(footer)

self.setCentralWidget(main_widget)

def crop_center(self, image, target_ratio=4/3):
    h, w = image.shape[:2]
    current_ratio = w / h

    if current_ratio > target_ratio:
        new_w = int(h * target_ratio)
        start_x = (w - new_w) // 2
        return image[:, start_x:start_x + new_w]
    else:
        new_h = int(w / target_ratio)
        start_y = (h - new_h) // 2
        return image[start_y:start_y + new_h, :]

def update_frame(self, frame):
    cropped = self.crop_center(frame, target_ratio=self.video_label.width() / self.video_label.height())
    rgb_image = cv2.cvtColor(cropped, cv2.COLOR_BGR2RGB)
    h, w, ch = rgb_image.shape
    bytes_per_line = ch * w
    qimg = QImage(rgb_image.data, w, h, bytes_per_line, QImage.Format_RGB888)

    self.video_label.setPixmap(QPixmap.fromImage(qimg).scaled(
        self.video_label.size(), Qt.KeepAspectRatio, Qt.SmoothTransformation))

def resizeEvent(self, event):
    super().resizeEvent(event)
    # Переміщуємо overlay_label
    self.overlay_label.resize(self.width() - 100, self.overlay_label.height())
    self.overlay_label.move(
        (self.width() - self.overlay_label.width()) // 2,
        80
    )

def closeEvent(self, event):
    self.recognizer.stop()
    event.accept()

```

Лістинг spectrogtam.py:

```

import numpy as np
import pyqtgraph as pg
from PyQt5.QtWidgets import QWidget, QVBoxLayout
from PyQt5.QtCore import QTimer, Qt
import sounddevice as sd

class AudioWaveformWidget(QWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setFixedHeight(200)

```

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

self.setStyleSheet("background-color: #1E1E2E;")

self.plot_widget = pg.PlotWidget()
self.plot_widget.setBackground('#1E1E2E')
self.plot_widget.setYRange(-100, 100)
self.plot_widget.setXRange(-1, 1)
self.plot_widget.setMouseEnabled(False, False)
self.plot_widget.hideAxis('bottom')
self.plot_widget.hideAxis('left')
self.plot_widget.setMenuEnabled(False)

layout = QVBoxLayout()
layout.setContentsMargins(0, 10, 0, 10)
layout.addWidget(self.plot_widget, alignment=Qt.AlignCenter)
self.setLayout(layout)

self.numBars = 60
half = self.numBars // 2
dx = 1.0 / half
self.x = np.array([i * dx for i in range(-half, half)])
self.data = np.zeros(self.numBars)

self.curves = []
for _ in range(self.numBars):
    curve = pg.PlotCurveItem(pen=pg.mkPen('#5A5AF7', width=20))
    self.plot_widget.addItem(curve)
    self.curves.append(curve)

self.timer = QTimer()
self.timer.timeout.connect(self.update_plot)
self.timer.start(30)

self.stream = sd.InputStream(callback=self.audio_callback, channels=1,
samplerate=44100, blocksize=self.numBars)
self.stream.start()

def audio_callback(self, indata, frames, time, status):
    if status:
        print(f"Stream warning: {status}")
    samples = indata[:self.numBars, 0]
    self.data = np.interp(samples, [-1, 1], [-2000, 2000]) # масштабу-
вання

def update_plot(self):
    amplitude = np.max(np.abs(self.data))
    color = pg.mkColor(255 * amplitude / 100, 0, 255)
    pen = pg.mkPen(color, width=2)

    for i, val in enumerate(self.data):
        x = self.x[i]
        y0 = 0
        y1 = val
        self.curves[i].setData([x, x], [y0, y1])
        self.curves[i].setPen(pen)

```

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг camera.py:

```
import cv2
from config import CAMERA_INDEX

def open_camera():
    cap = cv2.VideoCapture(CAMERA_INDEX)
    if not cap.isOpened():
        raise RuntimeError(f"[ERROR] Failed to open camera index {CAMERA_INDEX}")
    return cap
```

Лістинг detection.py:

```
from ultralytics import YOLO
import os
from config import YOLO_MODEL_PATH

def load_yolo_model():
    if not os.path.exists(YOLO_MODEL_PATH):
        raise FileNotFoundError(f"[ERROR] YOLO model not found: {YOLO_MODEL_PATH}")
    return YOLO(YOLO_MODEL_PATH)

def detect_faces(model, frame):
    results = model(frame, verbose=False)[0]
    boxes = results.boxes.xyxy.cpu().numpy() if results.boxes else []
    return boxes
```

Лістинг recognition.py:

```
from deepface import DeepFace
import cv2
import os
import numpy as np
from config import DB_FOLDER, DEEPFACE_MODEL, DETECTOR_BACKEND, FACE_IMG_SIZE

def prepare_database():
    if not os.path.exists(DB_FOLDER):
        raise FileNotFoundError(f"[ERROR] DB folder not found: {DB_FOLDER}")

    people = [name for name in os.listdir(DB_FOLDER)
               if os.path.isdir(os.path.join(DB_FOLDER, name))]

    if not people:
        raise RuntimeError(f"[ERROR] No people found in DB folder: {DB_FOLDER}")

    print(f"[INFO] Loaded {len(people)} people into face database:")
    for person in people:
        num_imgs = len([
            f for f in os.listdir(os.path.join(DB_FOLDER, person))
            if f.lower().endswith(('.jpg', '.jpeg', '.png'))
        ])
        print(f"      - {person} ({num_imgs} images)")

def preprocess_face(face_img):
    face = cv2.resize(face_img, FACE_IMG_SIZE)
```

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    return face

def recognize_face(face_img):
    try:
        face = preprocess_face(face_img)
        result = DeepFace.find(
            img_path=face,
            db_path=DB_FOLDER,
            enforce_detection=False,
            detector_backend=DETECTOR_BACKEND,
            model_name=DEEPFACE_MODEL
        )
        if not result[0].empty:
            identity_path = result[0].iloc[0]["identity"]
            person_name = os.path.basename(os.path.dirname(identity_path))
            return person_name
    except Exception as e:
        print(f"[WARN] Recognition error: {e}")
    return "Unknown"

```

Лістинг utils.py:

```

import cv2

def draw_label(frame, box, label, color):
    x1, y1, x2, y2 = map(int, box)
    cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
    cv2.putText(frame, label, (x1, y1 - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.6, color, 2)

```

Лістинг identification.py:

```

import time
import cv2
from PyQt5.QtCore import pyqtSignal, QThread

from camera import open_camera
from detection import load_yolo_model, detect_faces
from recognition import recognize_face, prepare_database
from utils import draw_label
from config import RECOGNITION_INTERVAL

from database import set_name

class Recognizer(QThread):
    frame_ready = pyqtSignal(object)

    def __init__(self, parent=None):
        super().__init__(parent)
        self.model = load_yolo_model()
        prepare_database()
        self.cap = open_camera()
        self.running = True
        self.last_recog_time = 0
        self.last_name = "Scanning..."

    def run(self):
        while self.running:

```

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

ret, frame = self.cap.read()
if not ret:
    break

boxes = detect_faces(self.model, frame)
for box in boxes:
    x1, y1, x2, y2 = map(int, box[:4])
    face = frame[y1:y2, x1:x2]

    now = time.time()
    if now - self.last_recog_time > RECOGNITION_INTERVAL:
        self.last_name = recognize_face(face)
        self.last_recog_time = now
        if self.last_name != "Unknown":
            set_name(self.last_name)

    color = (247, 90, 90) if self.last_name != "Unknown" else (0,
0, 255)
    draw_label(frame, (x1, y1, x2, y2), self.last_name, color)

    self.frame_ready.emit(frame)

def stop(self):
    self.running = False
    self.cap.release()
    self.quit()
    self.wait()

```

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг voice.py:

```

import sounddevice as sd
import torch
import torchaudio
import numpy as np
import tempfile
import asyncio
import random
from concurrent.futures import ThreadPoolExecutor
from speechbrain.pretrained import SpeakerRecognition

from database import get_voice_path_by_name

executor = ThreadPoolExecutor()

# model
recognizer = SpeakerRecognition.from_hparams(
    source="speechbrain/spkrec-ecapa-voxceleb",
    savedir="pretrained_models/spkrec"
)

# micro
def record_audio_to_file(duration=3, fs=16000):
    recording = sd.rec(int(duration * fs), samplerate=fs, channels=1,
dtype='float32')
    sd.wait()

    tmpfile = tempfile.NamedTemporaryFile(suffix=".wav", delete=False)
    waveform = torch.tensor(recording).T
    torchaudio.save(tmpfile.name, waveform, fs)
    return tmpfile.name

def compare_voices(file1, file2):
    score, prediction = recognizer.verify_files(file1, file2)
    return score.item(), prediction

async def main_voice():
    loop = asyncio.get_event_loop()

    def sync_voice_check():
        reference_file = get_voice_path_by_name()
        if not reference_file:
            return " ! Reference voice not found"

        recorded_file = record_audio_to_file(duration=3)
        score, is_same = compare_voices(reference_file, recorded_file)

        return True if is_same else False

    try:

```

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        result = await loop.run_in_executor(executor, sync_voice_check)
        return result
    except Exception as e:
        return f"❌ Помилка: {e}"

def generate_text():
    sentences = [
        "It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness.",
        "Call me Ishmael. Some years ago—never mind how long precisely—having little or no money in my purse.",
        "All happy families are alike; each unhappy family is unhappy in its own way.",
        "It is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want of a wife.",
        "In the beginning God created the heavens and the earth. Now the earth was formless and empty.",
        "There was no possibility of taking a walk that day. We had been wandering, indeed, in the leafless shrubbery.",
        "I am an invisible man. No, I am not a spook like those who haunted Edgar Allan Poe.",
        "It was a bright cold day in April, and the clocks were striking thirteen.",
        "The sun shone, having no alternative, on the nothing new.",
        "You don't know about me without you have read a book by the name of The Adventures of Tom Sawyer.",
        "It was a pleasure to burn. It was a special pleasure to see things eaten, to see things blackened.",
        "When he was nearly thirteen, my brother Jem got his arm badly broken at the elbow.",
        "I wish you to know that you have been the last dream of my soul.",
        "The sky above the port was the color of television, tuned to a dead channel.",
        "It was a queer, sultry summer, the summer they electrocuted the Rosenbergs."
    ]
    return random.choice(sentences)

```

		Загивий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг database.py:

```

import sqlite3

tmp_name = ''

def set_name(name):
    global tmp_name
    tmp_name = name

def get_surname():
    global tmp_name
    name = tmp_name
    conn = sqlite3.connect('security_database.db')
    cursor = conn.cursor()

    cursor.execute("SELECT surname FROM users WHERE name = ?", (name,))
    result = cursor.fetchone()
    conn.close()

    return result[0] if result else None

def get_name():
    global tmp_name
    name = tmp_name
    surname = get_surname()
    return name + ' ' + surname

def get_voice_path_by_name():
    global tmp_name
    name = tmp_name
    conn = sqlite3.connect('security_database.db')
    cursor = conn.cursor()

    cursor.execute("SELECT voice FROM users WHERE name = ?", (name,))
    result = cursor.fetchone()
    conn.close()

    return result[0] if result else None

def get_role_by_name():
    global tmp_name
    name = tmp_name
    conn = sqlite3.connect('security_database.db')
    cursor = conn.cursor()

    cursor.execute("SELECT role FROM users WHERE name = ?", (name,))
    result = cursor.fetchone()
    conn.close()

    return result[0] if result else None

def get_access_by_name():
    global tmp_name
    name = tmp_name
    conn = sqlite3.connect('security_database.db')
    cursor = conn.cursor()

```

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

cursor.execute("SELECT access FROM users WHERE name = ?", (name,))
result = cursor.fetchone()
conn.close()

return result[0] if result else None

```

Лістинг config.py:

```

DB_FOLDER = "security_db"
CAMERA_INDEX = 0
YOLO_MODEL_PATH = "yolov8n-face.pt"
RECOGNITION_INTERVAL = 2
DEEPFACE_MODEL = "Facenet"
DETECTOR_BACKEND = "opencv"
FACE_IMG_SIZE = (160, 160)

```

		Загрийвий А.С.			ККІК.КР.Б – 125 – 08 – ПЗ	Арк.
		Єфіменко А.А.				79
Змн.	Арк.	№ докум.	Підпис	Дата		