

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Система аналізу та впровадження біометричної
перевірки в процесі авторизації з використанням
нейронних мереж»**

Виконав студент групи ІПЗм-24-2
БАБИЦЬКИЙ Богдан Олександрович

Керівник роботи:
САВИЦЬКИЙ Роман Святославович

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«18» вересня 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: БАБИЦЬКИЙ Богдан Олександрович

Керівник роботи: САВІЦЬКИЙ Роман Святославович

Тема роботи: **«Система аналізу та впровадження біометричної перевірки в процесі авторизації з використанням нейронних мереж»**,

затверджена Наказом закладу вищої освіти від «18» вересня 2025 р., № 491/с

Вихідні дані для роботи: **веборієнтований сервіс для здійснення біометричної перевірки користувачів у процесі авторизації з використанням нейронних мереж**

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Савіцький Роман	22.09.2025р.	22.09.2025р.
2	Савіцький Роман	22.09.2025р.	22.09.2025р.
3	Савіцький Роман	22.09.2025р.	22.09.2025р.

РЕФЕРАТ

Кваліфікаційна робота освітнього ступеня «магістр» присвячена дослідженню та програмній реалізації системи біометричної верифікації в процесі авторизації користувачів засобами нейронних мереж.

Пояснювальна записка складається зі вступу, трьох розділів, висновків та списку використаної літератури. Пояснювальна записка до кваліфікаційної магістерської роботи містить 76 сторінки, 15 ілюстрацій, 1 таблицю та 30 посилань на зовнішні джерела.

Метою роботи є підвищення рівня захищеності інформаційних систем шляхом створення програмного модуля авторизації на основі розпізнавання облич.

У межах роботи розроблено вебсервіс для біометричної ідентифікації.

Наукова новизна дослідження полягає у визначенні оптимальної архітектури нейронної мережі для використання у веб-середовищі шляхом експериментального порівняння показників швидкодії та точності. Також було проаналізовано специфіку роботи алгоритму ArcFace на вибірках різної расової та гендерної приналежності, що дозволило виявити ступінь впливу демографічних факторів на якість верифікації та оцінити потенційну упередженість системи в реальних умовах експлуатації.

КЛЮЧОВІ СЛОВА: БІОМЕТРИЧНА ВЕРИФІКАЦІЯ, НЕЙРОННА МЕРЕЖА, ARCFACE, РОЗПІЗНАВАННЯ ОБЛИЧ, АВТОРИЗАЦІЯ, ВЕКТОРНИЙ АНАЛІЗ.

ABSTRACT

The Master's thesis is devoted to the research and software implementation of a biometric verification system in the user authorization process using neural networks.

The explanatory note consists of an introduction, three chapters, conclusions, and a list of references. The explanatory note to the Master's thesis contains 76 pages, 15 illustrations, 1 tables, and 30 references to external sources.

The aim of the work is to increase the security level of information systems by creating an authorization software module based on facial recognition.

The paper describes the process of creating a system that uses the ArcFace model to generate vector features (embeddings) and performs their comparative analysis. Methods for storing and searching vector data in relational databases have been investigated. Testing of verification accuracy and algorithm performance has been conducted.

KEYWORDS: BIOMETRIC VERIFICATION, NEURAL NETWORK, ARCFACE, FACE RECOGNITION, AUTHORIZATION, VECTOR ANALYSIS.

Зміст

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	11
1.1 Аналіз сучасних методів авторизації та проблеми безпеки доступу.....	11
1.2 Огляд та порівняльний аналіз архітектур нейронних мереж для розпізнавання облич	14
1.3 Постановка задачі дослідження	20
Висновки до розділу 1	22
РОЗДІЛ 2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ ТА ПРОЕКТУВАННЯ СИСТЕМИ.....	24
2.1 Огляд архітектур нейронних мереж для задач розпізнавання облич	24
2.2 Розробка алгоритму детекції та попередньої обробки зображень.....	28
2.3 Математична модель верифікації та оцінка подібності.....	31
Висновки до розділу 2.....	34
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ СИСТЕМИ.....	36
3.1 Обґрунтування вибору програмних та апаратних засобів реалізації	36
3.2 Експериментальне дослідження ефективності архітектур розпізнавання	41
3.3 Дослідження стійкості системи до демографічних факторів та аналіз алгоритмічної упередженості	45
3.4 Проектування архітектури клієнтського додатку та інтерфейсу користувача	47
Висновки до розділу 3	57
ВИСНОВКИ.....	59
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	61

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

ASGI – Asynchronous Server Gateway Interface

CNN – Convolutional Neural Networks

CPU – Central Processing Unit

FAR – False Acceptance Rate

FRR – False Rejection Rate

GDPR – General Data Protection Regulation

JWT – Json web token

JSON – JavaScript Object Notation

MTCNN – Multi-task Cascaded Convolutional Networks

GPU – Graphics Processing Unit

HNSW – Hierarchical Navigable Small World

OpenCV – Open Source Computer Vision Library

MVC – Model-View-Controller

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

R-Net – Refine Network

RBAC – Role-Based Access Control

ROC AUC – Receiver Operating Characteristic Area Under Curve

RGB – Red, Green, Blue

LWF – Labeled Faces in the Wild

ВСТУП

Актуальність теми. Стрімкий розвиток інформаційних технологій та глобалізація цифрового простору призвели до накопичення критичних обсягів даних в автоматизованих системах, що перетворює забезпечення їх конфіденційності та цілісності на першочергову проблему. Зростання обчислювальних потужностей та доступність інструментів злову призвели до того, що традиційні методи парольної аутентифікації вже не здатні гарантувати достатній рівень захисту. Вони стали вразливими до методів соціальної інженерії, фішингу та перехоплення, оскільки цілком залежать від людського фактору [1].

Сучасні методи комп'ютерного зору та глибокого навчання дозволили досягти безпрецедентної точності у задачах ідентифікації. Однак, із масовим впровадженням нейронних мереж виникає низка технічних та етичних проблем. По-перше, високоточні моделі часто є ресурсоемними, що ускладнює їх інтеграцію в системи реального часу або мобільні пристрої з обмеженою обчислювальною потужністю [2]. По-друге, існує проблема алгоритмічної упередженості, коли системи демонструють різну точність розпізнавання для різних демографічних груп (за ознаками статі, віку або етнічного походження), що є неприпустимим у критично важливих системах авторизації [3].

Таким чином, актуальність роботи полягає у необхідності створення такої системи біометричної верифікації, яка б поєднувала високу точність та швидкодію з оптимізованим використанням ресурсів, а також була б стійкою до демографічних факторів, мінімізуючи ймовірність помилкового доступу або відмови в обслуговуванні.

Мета роботи полягає у підвищенні ефективності та надійності процесу авторизації користувачів шляхом розробки системи біометричної верифікації на основі нейронних мереж, з обґрунтованим вибором архітектури за критерієм «швидкодія-точність» та мінімізацією алгоритмічної упередженості.

Завдання дослідження. Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Провести аналіз предметної області та існуючих методів біометричної аутентифікації, виявити їхні переваги та недоліки.
2. Дослідити сучасні архітектури згорткових нейронних мереж для задач розпізнавання облич.
3. Виконати порівняльний аналіз обраних моделей за показниками споживання системних ресурсів, швидкості обробки запитів та точності верифікації, обравши оптимальну для реалізації.
4. Спроекувати та розробити програмну систему верифікації, що включає серверну частину для обробки біометричних даних.
5. Провести експериментальне дослідження розробленої системи на наборах даних, що містять зображення людей різних національностей та статі, для виявлення та мінімізації можливої упередженості алгоритму.

Об'єктом дослідження є процес автоматизованої біометричної верифікації та авторизації користувачів в інформаційних системах.

Предметом дослідження є методи, моделі та програмні засоби глибокого навчання для розпізнавання облич, а також методи оцінки їх ефективності та стійкості до варіативності вхідних даних.

Методи дослідження. У роботі використано методи системного аналізу – для визначення вимог до системи; методи комп'ютерного зору та теорії штучних нейронних мереж – для реалізації модуля розпізнавання облич; методи експериментального дослідження – для порівняння продуктивності архітектур та оцінки точності роботи системи на різних вибірках даних.

Наукова новизна одержаних результатів полягає в удосконаленні методу порівняльного аналізу архітектур нейронних мереж для задач біометричної верифікації. На відміну від існуючих підходів, запропоновано використовувати комплексний критерій ефективності, що враховує не лише точність класифікації, а й експлуатаційні характеристики системи: час інференсу та споживання оперативної пам'яті. Це дозволило обґрунтовано визначити оптимальну модель ArcFace для роботи в умовах обмежених ресурсів вебсервера.

Також набули подальшого розвитку підходи до оцінювання надійності біометричних систем шляхом введення додаткового етапу тестування на демографічну упередженість. Це уможливило виявлення та кількісну оцінку відхилень у точності розпізнавання для різних етнічних та гендерних груп, що мінімізує ризики дискримінації при автоматизованому прийнятті рішень про допуск.

Практичне значення одержаних результатів. Розроблено програмну систему біометричної верифікації, яка може бути інтегрована як модуль авторизації у вебдодатки та корпоративні системи. Результати досліджень щодо швидкодії та упередженості дозволяють обґрунтовано обирати параметри моделі залежно від умов експлуатації, забезпечуючи баланс між безпекою та зручністю користування.

Публікації. За темою атестаційної магістерської роботи було опубліковано тези [4].

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз сучасних методів авторизації та проблеми безпеки доступу

В умовах стрімкого розвитку цифрових технологій захист доступу до персональних даних та корпоративних ресурсів стає першочерговим завданням. На практиці системи авторизації еволюціонували від простих механічних ключів до складних цифрових алгоритмів. Однак, попри наявність передових технологічних рішень, фундаментом більшості сучасних систем залишаються підходи, розроблені десятиліття тому, які вже не здатні забезпечити належний рівень захисту від актуальних загроз.

Домінуючим методом захисту доступу залишається автентифікація на основі знань, що реалізується через використання паролів та PIN-кодів. Широке застосування цього підходу зумовлене низькою вартістю інтеграції та відсутністю потреби у додатковому апаратному забезпеченні на стороні користувача. Парольний захист та вимоги до нього регламентуються міжнародними стандартами ISO/IEC 27002, який визначає правила обробки запам'ятовуваних секретів у вебсервісах та інформаційних системах. Проте, з точки зору безпеки, цей метод має критичні вразливості, пов'язані з «людським фактором». Користувачі схильні обирати прості комбінації для запам'ятовування (наприклад, дати народження), використовувати однакові паролі для різних облікових записів або записувати їх на ненадійних носіях. Крім того, пароль можна підглядіти, вгадати методом перебору або викрасти за допомогою фішингових атак [1].

Наступним щаблем розвитку стали апаратні засоби та методи підтвердження через сторонні канали зв'язку. До цієї категорії відносяться фізичні перепустки, смарт-карти, USB-токени, а також мобільні телефони, на які приходять одноразові коди доступу. Застосування апаратних засобів реалізує фактор володіння, що мінімізує ризики, притаманні виключно парольному захисту. Втім, використання фізичних носіїв створює загрозу їх фізичної компрометації внаслідок крадіжки або втрати, що унеможливорює подальшу авторизацію легітимного користувача. Додатковим вектором атак для методів,

що базуються на GSM-мережах, є можливість перехоплення трафіку, а також ризик відмови в обслуговуванні через відсутність стійкого мобільного покриття.

Обмеження традиційних механізмів захисту зумовили перехід до технологій, що базуються на факторі сутності [5]. Біометрична ідентифікація, на відміну від методів, заснованих на знанні (паролі) або володінні (токени), здійснюється шляхом автоматизованого вимірювання унікальних фізіологічних (відбитки пальців, геометрія обличчя, райдужна оболонка) або поведінкових характеристик користувача [1]. Такий підхід забезпечує жорстку прив'язку цифрового ідентифікатора до фізичної особи, що значно ускладнює підробку та унеможливорює передачу прав доступу третім особам (забезпечення невідомості дій) [6, 7].

Серед існуючих методів біометричної ідентифікації технологія розпізнавання обличчя демонструє значний потенціал для масового впровадження. На відміну від дактилоскопії, яка вимагає безпосереднього фізичного контакту зі сканером, технологія розпізнавання обличчя є повністю безконтактною. Така архітектура взаємодії виключає необхідність торкання сенсорів, що забезпечує гігієнічність процесу аутентифікації та суттєво підвищує пропускну здатність точок контролю доступу за рахунок скорочення часу на позиціонування користувача. Крім того, з економічної точки зору, такі системи часто не потребують спеціалізованого дороговартісного обладнання, оскільки можуть функціонувати на базі стандартних відеокамер високої роздільної здатності [8].

Практичне застосування систем розпізнавання обличчя ускладнюється впливом неконтрольованих умов експлуатації. Критичне зниження точності верифікації спричиняють такі фактори, як нерівномірність освітлення, геометричні спотворення внаслідок зміни ракурсу зйомки та часткове перекриття об'єкта. Класичні алгоритми комп'ютерного зору, що базуються на виділенні низькорівневих ознак (кутів, градієнтів, текстур), не забезпечують необхідної інваріантності до зазначених збурень [9]. Вони не здатні ефективно узагальнювати інформацію про обличчя при зміні умов освітлення або повороті

голови, що призводить до високого рівня помилок першого та другого роду. Класичні методи, які покладаються на фіксовані геометричні шаблони або антропометричні точки, виявляються безсилими перед природною варіативністю людської зовнішності у динамічному середовищі. Це зумовлює необхідність застосування більш досконалих математичних моделей, здатних до адаптивного навчання.

Вирішенням проблеми нестійкості до варіативності вхідних даних стало застосування методів глибокого навчання, зокрема згорткових нейронних мереж (CNN). Сучасні архітектури, такі як ArcFace, FaceNet та VGG Face, дозволяють формувати векторні представлення облич, які є інваріантними до змін умов зйомки. Глибокі нейронні мережі здатні виділяти високорівневі абстрактні ознаки, що описують унікальну структуру обличчя, ігноруючи шум та незначні зміни у зовнішності. На відміну від попередніх поколінь алгоритмів, нейронні мережі не потребують ручного конструювання ознак, а самостійно виявляють найбільш дискримінативні характеристики в процесі навчання на великих масивах даних. Саме порівняльний аналіз цих архітектур та вибір оптимальної моделі є ключовим завданням для створення надійної системи авторизації.

Разом із масовим впровадженням нейромережевих технологій виникає низка нових технічних та етичних викликів. По-перше, високоточні моделі глибокого навчання часто є ресурсоемними, що ускладнює їх інтеграцію в системи реального часу або мобільні пристрої з обмеженою обчислювальною потужністю. Глибокі архітектури, що містять мільйони параметрів, вимагають значних обчислювальних ресурсів процесора (CPU) або графічного прискорювача (GPU), а також значного обсягу оперативної пам'яті. Це вимагає пошуку балансу між точністю розпізнавання та швидкістю роботи алгоритмів, оскільки занадто велика затримка при обробці запиту на авторизацію негативно впливає на користувацький досвід та пропускну здатність системи.

По-друге, існує проблема алгоритмічної упередженості, коли системи демонструють різну точність розпізнавання для різних демографічних груп. Дослідження показують, що нейронні мережі, навчені на незбалансованих

наборах даних, можуть демонструвати суттєво гірші показники точності при ідентифікації представників певних етнічних груп, статі або вікових категорій. Похибки у роботі алгоритмів за ознаками статі, віку або етнічного походження є неприпустимими у критично важливих системах авторизації, оскільки це може призвести до дискримінації користувачів або безпідставної відмови у доступі легітимним особам.

Таким чином, актуальність подальших досліджень полягає у необхідності створення такої системи біометричної верифікації, яка б поєднувала високу точність та швидкодію з оптимізованим використанням ресурсів, а також була б стійкою до демографічних факторів. Важливим аспектом є забезпечення стійкості системи до демографічних факторів, що дозволить мінімізувати ймовірність помилкового доступу або безпідставної відмови в обслуговуванні для будь-яких категорій користувачів. Розробка ефективного механізму авторизації вимагає комплексного підходу, що включає не лише вибір архітектури нейронної мережі, але й оптимізацію процесів попередньої обробки даних та налаштування метрик порівняння. Порівняльний аналіз сучасних архітектур нейронних мереж та вибір оптимальної моделі є ключовим етапом для створення надійної та безпечної системи авторизації, що відповідає сучасним стандартам цифрової ідентифікації.

1.2 Огляд та порівняльний аналіз архітектур нейронних мереж для розпізнавання облич

Фундаментальним завданням нейронної мережі в сучасних системах біометричної верифікації є перетворення вхідного растрового зображення обличчя у компактний вектор числових значень, який у професійній літературі називають ембедінгом. Формально цю задачу можна описати як пошук такої складної нелінійної функції відображення $F(x)$, яка трансформує вхідний багатовимірний масив пікселів x у вектор ознак v значно меншої розмірності. Математичний зміст цього перетворення полягає у проєкції даних з високовимірному простору пікселів (де розмірність визначається добутком ширини, висоти та кількості каналів зображення) у низьковимірний метричний

простір, зберігаючи при цьому семантичну інформацію про ідентичність об'єкта. Суть формування такого вектора полягає у кодуванні унікальних біометричних ознак людини та одночасному відсіюванні шумових факторів, таких як варіації освітлення, зміна ракурсу, фон чи мімічні прояви.

Головна мета процесу навчання полягає в досягненні такого розміщення векторів у багатовимірному просторі ознак, де вектори, отримані з різних зображень однієї особи, будуть знаходитись максимально близько один до одного (мінімізація внутрішньокласової дисперсії). Водночас вектори, що відповідають зображенням різних людей, повинні бути значно віддалені (максимізація міжкласової відстані). Ця вимога є критичною для забезпечення надійності системи, адже в такому метричному просторі евклідова або кутова відстань між векторами виступає об'єктивним критерієм схожості при прийнятті автоматизованого рішення про верифікацію користувача. Ефективна кластеризація векторів дозволяє нівелювати вплив зовнішніх умов зйомки та фокусуватися виключно на інваріантних характеристиках особистості.

На сьогодні для вирішення цього завдання найефективнішим інструментом є глибокі згорткові нейронні мережі (CNN). Їхній розвиток дозволив здійснити перехід від традиційних алгоритмів класифікації [10], де модель обмежена фіксованим набором відомих облич (closed-set identification), до побудови універсальних метричних систем. Такі системи здатні узагальнювати вивчені закономірності та коректно розпізнавати людей, зображення яких були відсутні у навчальній вибірці. Це забезпечує високу точність розпізнавання у режимі відкритої множини (open-set recognition), що є необхідною умовою для масштабування сучасних систем контролю доступу без необхідності перенавчання моделі при реєстрації кожного нового користувача.

Процедура розпізнавання не виконується безпосередньо на вихідних пікселях зображення, оскільки їх числові значення мають високу кореляцію з умовами освітлення та положенням камери, а не з біометричними рисами. Нейронна мережа реалізує поетапну трансформацію вхідних даних, застосовуючи каскад операцій згортки (convolution), нелінійної активації та

підвибірки (pooling). Операція згортки забезпечує виокремлення локальних інформативних ознак шляхом застосування набору фільтрів (ядер), вагові коефіцієнти яких оптимізуються у процесі навчання. Важливим етапом є застосування нелінійних функцій активації, що дозволяє моделі апроксимувати складні залежності у даних, які неможливо описати лінійними методами. У свою чергу, операції підвибірки поступово зменшують просторову розмірність карт ознак, що знижує обчислювальну складність алгоритму та забезпечує інваріантність системи до незначних геометричних зсувів і масштабування об'єкта в кадрі.

Процес вилучення ознак у глибоких мережах побудовано ієрархічно, що концептуально імітує принципи обробки візуальної інформації зоровою корою головного мозку людини. На початкових рівнях архітектури система фіксує прості геометричні примітиви та низькорівневі патерни, такі як лінії, орієнтовані грані, кути та контрастні переходи кольорів. На середніх шарах ці розрізнені примітиви комбінуються, утворюючи локальні частини обличчя (очі, ніс, губи) та їх взаємне розташування. На найглибших рівнях мережі відбувається інтеграція локальних ознак у складні семантичні структури, що формують цілісний, абстрактний образ особи. Результатом цього багатоступеневого аналізу стає генерація компактного цифрового дескриптора – вектора ознак фіксованої довжини (найчастіше 128, 256 або 512 чисел). Для коректного порівняння отримані вектори зазвичай нормалізуються, що призводить до їх розміщення на поверхні багатовимірної гіперсфери, спрощуючи обчислення ступеня подібності через косинус кута між ними.

Однією з перших архітектур, яка продемонструвала можливість досягнення точності розпізнавання, порівнянної з людською, стала модель VGG-Face, розроблена групою дослідників Visual Geometry Group з Оксфордського університету. Ця архітектура базується на концепції суттєвого збільшення глибини нейронної мережі (до 16 або 19 шарів) при використанні фільтрів згортки малого розміру (3 x 3 пікселі). Така будова дозволяє ефективно виділяти ієрархічні ознаки зображення без надмірного ускладнення операцій на кожному

окремому шарі, забезпечуючи при цьому високу нелінійність моделі, що є важливим для розпізнавання складних патернів обличчя [10].

Структурно мережа складається з послідовних блоків згорткових шарів, розділених операціями субдискретизації (max-pooling), які поступово зменшують просторову розмірність карт ознак, одночасно збільшуючи їхню глибину. Завершується архітектура трьома повнозв'язними шарами, перші два з яких містять по 4096 нейронів, а останній виконує класифікацію. Саме наявність масивних повнозв'язних шарів зумовлює головний недолік цієї моделі – величезну кількість параметрів (близько 138 мільйонів), що призводить до значного споживання оперативної пам'яті та великого розміру файлу ваг на диску.

Важливим аспектом розробки VGG-Face стало створення спеціалізованого методичного підходу до навчання. Оскільки на момент створення моделі публічні датасети були недостатньо великими для тренування такої глибокої мережі без перенавчання, автори застосували процедуру автоматичного збору та фільтрації зображень з інтернету. Це дозволило сформувати набір даних, що містив 2,6 мільйона зображень для 2622 осіб. Використання такої масштабної навчальної вибірки у поєднанні з функцією активації ReLU (Rectified Linear Unit) дозволило досягти високої стійкості розпізнавання, хоча й ціною високих обчислювальних витрат [10, 11].

Тим не менш, VGG Face залишається важливим еталоном у наукових дослідженнях. Її стабільність та передбачуваність дозволяють використовувати її як точку відліку для оцінки ефективності новіших, більш легковагих архітектур, які намагаються досягти тієї ж точності при менших обчислювальних затратах.

Важливим етапом еволюції методів розпізнавання стало впровадження підходів, що базуються на глибокому метричному навчанні (Deep Metric Learning). Яскравим представником цього напрямку є архітектура FaceNet, запропонована дослідниками Google. На відміну від попередніх моделей, які використовували проміжні шари класифікаторів (наприклад, Softmax) і

розглядали вектор ознак як побічний продукт навчання, FaceNet навчається відображати зображення безпосередньо у компактний евклідов простір. Результатом роботи такої мережі є вектор ознак фіксованої розмірності (зазвичай 128 байт), який безпосередньо репрезентує ступінь схожості облич. Основна гіпотеза методу полягає в тому, що у сформованому багатовимірному просторі евклідова відстань відповідає семантичній схожості: вектори зображень однієї особи групуються компактно, тоді як вектори різних людей значно віддалені один від одного [8].

Основою навчання моделі FaceNet є застосування спеціалізованої функції втрат Triplet Loss. Цей математичний підхід базується на одночасному опрацюванні трійок зображень: якірного зразка (Anchor), позитивного прикладу (Positive – інше фото тієї ж особи) та негативного прикладу (Negative – зображення сторонньої людини). Головне завдання алгоритму оптимізації – налаштувати ваги мережі таким чином, щоб скоротити відстань між якірним і позитивним векторами, і водночас віддалити негативний вектор на величину, що перевищує заданий фіксований відступ (margin). Це дозволяє не просто класифікувати обличчя, а структурувати сам простір ознак, забезпечуючи високу розрізняльну здатність системи.

Попри високу теоретичну ефективність, метод Triplet Loss має суттєві практичні недоліки, що обмежують його застосування. Процес навчання є обчислювально складним і повільним через необхідність ретельного підбору так званих «важких» трійок (hard triplets) – наборів даних, де мережа помиляється найчастіше. Якщо використовувати випадкові трійки, градієнти швидко стають близькими до нуля, і навчання зупиняється. Водночас надмірна концентрація на найскладніших прикладах може призвести до нестабільності збіжності моделі та спотворення простору ознак. Ця чутливість до стратегії формування навчальної вибірки (mining strategy) робить налаштування FaceNet значно складнішим порівняно з класифікаційними підходами [7].

Найбільш досконалою серед проаналізованих моделей є архітектура ArcFace, яка представляє сучасний етап еволюції алгоритмів глибокого навчання

для задач біометричної ідентифікації. Ця архітектура була розроблена для подолання фундаментальних обмежень попередніх підходів, зокрема нестабільності навчання методів на основі євклідової відстані (таких як Triplet Loss) та недостатньої дискримінативної здатності класичних функцій активації. Ключовою інновацією ArcFace є зміна парадигми метричного навчання: замість оптимізації відстаней у плоскому просторі, модель виконує відображення векторів ознак на поверхню багатовимірної гіперсфери фіксованого радіуса. Для покращення роздільної здатності між класами та формування чітких меж прийняття рішень використовується спеціалізована функція втрат Additive Angular Margin Loss [12].

Математична сутність методу полягає у введенні додаткового штрафного коефіцієнта – адитивного кутового відступу – до цільового кута між нормалізованим вектором ознак та вектором ваг відповідного класу. У процесі навчання алгоритм примусово збільшує кутову відстань між класами, додаючи фіксоване значення відступу до аргументу косинусної функції. Це змушує нейронну мережу генерувати такі ембедінги, які не лише коректно класифікуються, але й розташовуються максимально близько до центру свого кластера. Такий підхід гарантує мінімізацію внутрішньокласової дисперсії (компактність ознак однієї особи) та одночасну максимізацію міжкласової відстані (сепарабельність різних осіб) у геодезичному просторі. На практиці застосування ArcFace забезпечує системі високу інваріантність до складних умов зйомки, нівелюючи вплив таких факторів, як зміна ракурсу голови, вікові трансформації обличчя або нерівномірне освітлення [4, 13].

Узагальнюючи результати проведеного порівняльного аналізу, можна констатувати, що кожна з розглянутих архітектур має специфічні особливості, які визначають сферу їхнього доцільного застосування. Модель VGG-Face, хоч і демонструє високі показники точності, характеризується критичним недоліком – надмірним споживанням оперативної пам'яті, що робить її малопридатною для розгортання в системах реального часу з обмеженими апаратними ресурсами. Архітектура FaceNet, попри свою концептуальну значущість та компактність

вихідних даних, виявилася найскладнішою в налаштуванні через нестабільність процедури навчання, а також продемонструвала найнижчу швидкість обробки запитів серед протестованих зразків.

Натомість архітектура ArcFace забезпечує найкращий баланс між обчислювальною ефективністю та якістю роботи. Завдяки використанню вдосконаленої математичної моделі з кутовими відступами, вона дозволяє досягти високої точності ідентифікації при помірному навантаженні на систему. Сукупність цих технічних характеристик, а також доведена стійкість алгоритму до варіативності вхідних даних, стала підставою для обрання ArcFace як базової архітектури для практичної реалізації модуля біометричної верифікації в даній роботі [4].

1.3 Постановка задачі дослідження

Проведений у попередніх підрозділах аналіз сучасних методів авторизації дозволив виявити стійку тенденцію до переходу від традиційних парольних систем до біометричних засобів захисту. Це зумовлено тим, що класичні методи, описані в стандартах цифрової ідентифікації, все частіше демонструють вразливість до атак соціальної інженерії та фішингу [5]. Зокрема, системи розпізнавання обличчя демонструють найкращі показники зручності для користувача та санітарної безпеки, оскільки забезпечують безконтактну взаємодію [6].

Однак, огляд існуючих нейромережевих архітектур показав, що універсального рішення не існує: кожна модель має свої обмеження щодо обчислювальної складності або стійкості до варіативності умов зйомки [8]. Критичною проблемою залишається висока ресурсоемність глибоких згорткових мереж (CNN). Багато сучасних архітектур, що демонструють рекордні показники точності на академічних датасетах, вимагають значних обчислювальних потужностей, що ускладнює їх інтеграцію у вебдодатки та мобільні системи без використання дороговартісного спеціалізованого обладнання [2].

Окремим викликом є проблема алгоритмічної упередженості (bias). Дослідження вказують на те, що популярні нейронні мережі часто демонструють

нерівномірну точність розпізнавання для різних демографічних груп, що пов'язано з дисбалансом навчальних вибірок [3]. У контексті систем безпеки та авторизації така дискримінація є неприпустимою, оскільки вона може призводити як до помилкових відмов у доступі легітимним користувачам, так і до зниження рівня захищеності системи загалом.

Головною науково-технічною проблемою, що розглядається в роботі, є необхідність створення системи, яка забезпечувала б високу точність верифікації в умовах реальної експлуатації, залишаючись при цьому достатньо швидкодіючою для роботи в режимі реального часу та стійкою до демографічних факторів. Результати попередніх експериментальних досліджень авторів вказують на те, що найбільш перспективним підходом є використання архітектури ArcFace, яка завдяки функції втрат з адитивним кутовим відступом дозволяє досягти кращої сепарації класів порівняно з аналогами [4, 13].

Виходячи з цього, метою магістерської роботи є підвищення надійності та ефективності процедури авторизації користувачів в інформаційних системах шляхом розробки та впровадження програмного комплексу біометричної верифікації на основі глибоких нейронних мереж.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. Провести аналіз предметної області та обґрунтувати вибір методів біометричної ідентифікації.
2. Дослідити ефективність сучасних архітектур згорткових нейронних мереж та обрати оптимальну модель для вирішення задачі верифікації.
3. Розробити алгоритмічне та програмне забезпечення системи авторизації.

Для вирішення поставлених завдань застосовано комплекс взаємопов'язаних методів. Теоретичний аналіз та узагальнення дозволили систематизувати існуючі підходи до біометричної ідентифікації [6, 8], а методи системного аналізу – сформулювати вимоги та побудувати модель системи авторизації згідно зі стандартами цифрової ідентифікації [5]. Безпосередня реалізація, що включає детекцію облич за алгоритмом MTCNN та генерацію векторних ознак, ґрунтується на методах комп'ютерного зору та глибокого

навчання [2, 14]. Для оцінки результатів використано емпіричні методи та математичну статистику, за допомогою яких проведено порівняльний аналіз архітектур VGG-Face та ArcFace [4], тестування на демографічну стійкість на базі FairFace [15] та розрахунок метрик якості класифікації інструментами Scikit-learn [16].

Таким чином, у першому розділі виконано формалізацію задачі дослідження та обґрунтовано доцільність переходу від традиційних парольних систем до біометричних методів ідентифікації. На основі критичного аналізу існуючих технологій визначено, що пріоритетним інструментом для розв'язання поставленої проблеми є апарат глибоких згорткових нейронних мереж. Зокрема, вибір архітектури ArcFace обумовлений її здатністю забезпечувати високу метричну точність верифікації при збереженні прийнятних показників швидкодії. Визначені критерії ефективності та сформульовані вимоги до стійкості алгоритму щодо демографічних факторів складають теоретичне підґрунтя для подальшої практичної реалізації системи, технічні аспекти якої будуть розкриті у наступних розділах роботи.

Висновки до розділу 1

У першому розділі магістерської роботи здійснено комплексний аналіз проблеми забезпечення інформаційної безпеки та контролю доступу в сучасних автоматизованих системах. Результати дослідження засвідчили, що традиційні підходи, такі як парольний захист та апаратні токени, мають критичні вразливості, зумовлені впливом людського фактору та ризиками фізичної втрати носіїв. У цьому контексті обґрунтовано, що перехід до біометричних систем, зокрема технології розпізнавання обличчя, є безальтернативним шляхом підвищення надійності та зручності процедури верифікації.

Водночас визначено ключові фактори, що обмежують ефективність біометричних систем у реальних умовах експлуатації: залежність точності від неконтрольованого освітлення, варіативності ракурсу зйомки та наявності оклюзій. Встановлено, що класичні методи комп'ютерного зору не забезпечують

необхідної інваріантності до цих збурень, що зумовлює необхідність застосування глибоких згорткових нейронних мереж.

На основі порівняльного аналізу сучасних архітектур, зокрема VGG Face, FaceNet та ArcFace, доведено переваги останньої. З'ясовано, що завдяки використанню функції втрат з адитивним кутовим відступом модель ArcFace демонструє найкращу сепарацію класів та стійкість до варіативності вхідних даних при суттєво меншій обчислювальній складності порівняно з аналогами.

Виходячи з виявлених проблем та обраного математичного апарату, сформульовано постановку задачі дослідження. Метою роботи визначено розробку системи біометричної верифікації на базі ArcFace, що дозволить підвищити ефективність процедури авторизації. Також визначено об'єкт, предмет, наукову новизну та практичне значення результатів, що створює теоретичне підґрунтя для подальшої практичної реалізації системи.

Результати, отримані в першому розділі, є теоретичним підґрунтям для розробки алгоритмічного забезпечення та програмної реалізації системи, що буде розглянуто в наступних розділах роботи.

РОЗДІЛ 2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ ТА ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Огляд архітектур нейронних мереж для задач розпізнавання облич

У сучасних системах комп'ютерного зору задача розпізнавання облич зазнала фундаментальних змін завдяки переходу від використання евристичних алгоритмів (наприклад, геометричних шаблонів або локальних бінарних шаблонів) до методів глибокого навчання. При проектуванні системи біометричної верифікації ключовим завданням є вибір та обґрунтування математичної моделі, здатної трансформувати високовимірні вхідні дані (зображення) у компактний та інформативний набір числових значень.

Формально цю задачу можна описати як пошук складної нелінійної функції відображення $F(x)$, яка перетворює вхідне фото x (тензор пікселів) у вектор ознак (ембедінг) v фіксованої розмірності. Процес навчання такої архітектури спрямований на побудову багатовимірного метричного простору, топологія якого відображає семантичну схожість об'єктів.

Головна вимога до цього вектора є метричною: евклідова або кутова відстань між векторами фотографій однієї людини має бути мінімальною (внутрішньокласова компактність), а між векторами різних людей – максимальною (міжкласова сепарабельність) [2]. Виконання цієї умови є критичним, оскільки саме на основі розрахованої дистанції приймається бінарне рішення про допуск або відмову в авторизації.

Еволюція архітектур нейронних мереж у цьому напрямку відбувалася шляхом поглиблення структури та вдосконалення функцій втрат. Якщо ранні моделі фокусувалися на задачі класифікації на фіксованій множині облич, то сучасні підходи, такі як FaceNet та ArcFace, орієнтовані на пряму оптимізацію відстаней у просторі ознак. Це дозволяє створювати системи, стійкі до варіативності освітлення, ракурсу та вікових змін. У даному підрозділі проаналізовано ключові архітектурні рішення, що стали стандартами в індустрії, та визначено їхні переваги й недоліки в контексті розроблюваної системи.

В ході дослідження проаналізовано еволюцію підходів: від ранніх методів класифікації до сучасного метричного навчання. Знаковим представником першої групи є архітектура VGG Face. В основі її роботи лежить принцип класифікації, де модель навчається розрізняти людей із заздалегідь визначеного набору. Математично це завдання вирішується за допомогою функції Softmax, яка трансформує вихідні сигнали мережі у розподіл ймовірностей, визначаючи належність вхідного зображення до одного з відомих системі класів. Для конкретної людини (класу i) ймовірність P_i розраховується як відношення експоненти її показника до суми експонент усіх можливих людей у вибірці (2.1):

$$P_i = \frac{e^{z_i}}{\sum_{j=1}^N e^{z_j}} \quad (2.1)$$

де:

z_i – значення «впевненості» мережі для класу i ;

N – загальна кількість класів (людей), на яких навчалася мережа;

P_i – підсумкова ймовірність (значення в діапазоні від 0 до 1) [10, 17].

Головний недолік методу Softmax у чистому вигляді полягає в тому, що він оптимізує ймовірність правильної класифікації, але не гарантує компактності векторів у просторі ознак. Це означає, що коли системі потрібно верифікувати людину, якої не було в навчальній базі, точність розпізнавання суттєво знижується, оскільки модель не навчена ефективно розділяти «незнайомі» обличчя.

Наступним етапом розвитку стала модель FaceNet, яка відмовилася від проміжної класифікації на користь прямого метричного навчання. Замість визначення конкретної особи, мережа навчається розміщувати вектори схожих обличчя поруч у багатовимірному евклідовому просторі, а різних – на віддалі.

Основою FaceNet є функція втрат Triplet Loss (втрати на трійках). Навчання відбувається на трьох зображеннях одночасно, таких як якір (A) – базове зображення певної людини, позитивний приклад (P) – інше зображення тієї ж людини, негативний приклад (N) – зображення іншої людини.

Мета навчання полягає в тому, щоб відстань між векторами «своїх» була меншою, ніж відстань до «чужих», з урахуванням певного запасу (маржі). Ця умова математично записується нерівністю (2.2):

$$|F(A) - F(P)|^2 + \alpha < |F(A) - F(N)|^2 \quad (2.2)$$

де:

$|\cdot|^2$ – квадрат евклідової відстані між векторами ознак;

α – маржа, параметр, який задає мінімально необхідну різницю між позитивною та негативною парами [6, 8].

Використання Triplet Loss дозволило значно покращити результати верифікації порівняно з класифікаційними підходами. Однак, з математичної та практичної точки зору, такий підхід є складним у реалізації: ефективність навчання критично залежить від алгоритму підбору «важких» трійок – прикладів, на яких мережа помиляється найчастіше. Це робить процес тренування моделі повільним та потенційно нестабільним [12].

Якщо попередні архітектури намагалися розсунути вектори у звичайному плоскому просторі, то модель ArcFace, обрана в даній роботі, використовує принципово інший геометричний підхід. Вона базується на припущенні, що обличчя найкраще описувати не координатами, а кутами. Це логічно, оскільки перед порівнянням усі вектори ознак нормалізуються, тобто приводяться до фіксованої довжини. У такому випадку всі вони лежать на поверхні багатовимірної гіперсфери.

Математичною основою ArcFace є модифікована функція втрат Additive Angular Margin Loss.

У звичайних мережах алгоритм просто намагається зробити вектор схожим на еталон класу. ArcFace ускладнює це завдання, додаючи до кута між вектором і центром класу штучний «штраф» – відступ m .

Спрощено основну ідею розрахунку оцінки (score) можна записати формулою (2.3):

$$Score = \cos(\theta + m) \quad (2.3)$$

де:

θ – кут між вектором конкретного зображення та центром класу цієї людини (чим менший кут, тим точніша відповідність);

m – додатковий кут.

Цей параметр m діє як бар'єр. Він змушує нейронну мережу «стискати» сектор, у який потрапляють варіації обличчя однієї людини. Це створює чітку порожню зону (зазор) між кластерами різних людей на сфері, що значно знижує ймовірність помилкової ідентифікації.

Повна функція втрат для одного зразка виглядає наступним чином (2.4):

$$L = -\log \frac{e^{s \cdot \cos(\theta_{y_i} + m)}}{e^{s \cdot \cos(\theta_{y_i} + m)} + \sum_{j \neq y_i} e^{s \cdot \cos(\theta_j)}} \quad (2.4)$$

Пояснення параметрів моделі:

s – масштаб сфери. Це технічний параметр, який розтягує простір ознак, роблячи процес навчання стабільнішим;

m – ключовий параметр. Саме завдяки додаванню m мережа вчиться розпізнавати обличчя навіть у складних умовах (поворот голови, погане освітлення), оскільки вона формує «запас міцності» при розрахунках [12].

Застосування такого геометричного підходу дозволяє моделі ArcFace формувати найбільш дискримінативні ознаки порівняно з розглянутими аналогами. Якщо архітектура VGG-Face вимагає значних обчислювальних ресурсів через велику кількість параметрів, а FaceNet характеризується складністю підбору навчальних трійок, то ArcFace забезпечує оптимальне співвідношення між точністю ідентифікації та швидкістю обробки запитів. Саме здатність до ефективної кластеризації векторів на гіперсфері, забезпечена функцією втрат Additive Angular Margin Loss, робить цю архітектуру пріоритетним вибором для практичної реалізації системи біометричної верифікації, де критичними вимогами є надійність розпізнавання та робота в режимі реального часу [4, 12].

2.2 Розробка алгоритму детекції та попередньої обробки зображень

Критичною умовою забезпечення високої точності верифікації є якісна попередня нормалізація вхідних даних, оскільки безпосередня подача на вхід нейронної мережі «сирих» кадрів відеопотоку є неефективною [1]. Це зумовлено впливом неконтрольованих умов зйомки: наявністю складного фону (шумові артефакти), варіативністю масштабу обличчя та геометричними спотвореннями, спричиненими довільним ракурсом [2]. Такі фактори унеможливають коректне формування вектора ознак моделлю ArcFace, яка вимагає суворо фіксованої розмірності вхідного тензора (112×112 пікселів) та канонічного положення об'єкта [3].

Для нівелювання зазначених впливів у системі імплементовано багатоступеневий конвеєр попередньої обробки, побудований на базі архітектури MTCNN (Multi-task Cascaded Convolutional Networks) [4]. Вибір цього методу зумовлений його архітектурною специфікою – використанням каскаду глибоких мереж, що дозволяє виконувати мультизадачне навчання [5]. Алгоритм не лише локалізує обличчя в кадрі, відсіюючи фонові об'єкти, а й одночасно реалізує регресію координат п'яти ключових антропометричних точок (очей, носа та кутів рота) [6]. Наявність цих координат є необхідною умовою для подальшого геометричного вирівнювання зображення шляхом афінних перетворень, що гарантує горизонтальне положення лінії очей незалежно від нахилу голови користувача [7].

Фінальним етапом підготовки даних є процедура нормалізації яскравості, яка приводить значення пікселів до єдиного діапазону (від -1 до 1), центрує дані та покращує збіжність нейромережі [8]. Застосування такої комплексної процедури гарантує, що на вхід моделі розпізнавання завжди подаються стандартизовані дані, що мінімізує помилки верифікації та підвищує загальну надійність системи [9].

Для нівелювання зазначених впливів у системі імплементовано конвеєр попередньої обробки на базі архітектури MTCNN. Вибір цього методу зумовлений його архітектурною специфікою – використанням каскаду глибоких

мереж, що дозволяє виконувати мультизадачне навчання: одночасно реалізувати регресію координат обмежувальної рамки та високоточну локалізацію п'яти ключових антропометричних точок для подальшого афінного вирівнювання зображення [8].

Робота MTCNN базується на концепції каскадної обробки та використання піраміди зображень.

Перед початком аналізу вхідне зображення I масштабується кілька разів, створюючи набір копій різного розміру. Це необхідно для забезпечення інваріантності до масштабу: алгоритм має однаково успішно знаходити як великі обличчя (поблизу камери), так і малі (на відстані).

Архітектурна реалізація MTCNN базується на принципі каскадної фільтрації, що передбачає послідовну обробку даних трьома взаємопов'язаними нейронними мережами зростаючої складності. На початковому етапі мережа P-Net виконує сканування піраміди зображень для генерації множини вікон-кандидатів, відсіюючи очевидні фонові об'єкти. Отримані області інтересу передаються на вхід мережі R-Net, яка здійснює більш глибоку фільтрацію хибнопозитивних спрацьовувань та уточнення координат рамок із застосуванням алгоритму придушення не максимумів. Фінальний етап, реалізований мережею O-Net, забезпечує високоточну локалізацію обличчя та регресію координат п'яти ключових антропометричних точок, необхідних для подальшого геометричного вирівнювання [6, 8].

Застосування такого три етапного підходу дозволяє системі ігнорувати складний фон і передавати на вхід моделі розпізнавання лише чітко локалізовану область інтересу.

Важливою особливістю алгоритму MTCNN, що відрізняє його від класичних методів детекції [9], є використання мульти-задачного навчання. Це означає, що кожна з трьох мереж каскаду навчається вирішувати три різні задачі одночасно, що дозволяє досягти синергетичного ефекту в точності розпізнавання.

Математична модель навчання будується на мінімізації об'єднаної функції втрат. Розглянемо складові цієї функції детальніше. Для кожного вхідного зразка x_i мережа повинна визначити ймовірність того, що це зображення є обличчям.

Для розв'язання задачі класифікації «Обличчя / Не обличчя» застосовується функція втрат перехресної ентропії. Вона дозволяє оцінити розбіжність між передбаченою мережею ймовірністю наявності обличчя та істинною міткою класу, штрафуючи модель за впевнені, але помилкові передбачення.

Паралельно з цим виконується регресія обмежувальної рамки. У цьому випадку мережа вчиться передбачати вектор зміщення відносно запропонованого вікна. Для мінімізації помилки позиціювання використовується евклідова відстань, яка обчислює різницю між прогнозованими параметрами рамки та її реальними координатами. За аналогічним принципом працює і локалізація п'яти антропометричних точок (очей, носа та кутів рота), мережа мінімізує відхилення передбачених координат від еталонних.

Загальна функція втрат, яку оптимізує детектор під час навчання, є зваженою сумою похибок класифікації, регресії рамки та локалізації точок. Варто зазначити, що вагові коефіцієнти для кожної складової не є статичними і варіюються залежно від стадії каскаду. Наприклад, для фінальної мережі O-Net критично важливою є точність розміщення ключових точок, тому внесок відповідної помилки у загальну функцію втрат встановлюється вищим, ніж на етапі P-Net.

Наступним важливим кроком є геометричне вирівнювання обличчя. Оскільки навіть точно детектоване обличчя може мати нахил, що негативно впливає на якість розпізнавання моделлю ArcFace, застосовується перетворення подібності. Алгоритм використовує знайдені координати п'яти точок і зіставляє їх з «канонічним» шаблоном, де лінія очей є горизонтальною. На основі цього розраховується матриця афінного перетворення, що включає необхідний кут повороту, коефіцієнт масштабування та зсув. Застосування цієї матриці до

вхідного зображення дозволяє отримати стандартизований та вирівняний зразок обличчя.

Застосування такої процедури гарантує, що на вхід нейронної мережі розпізнавання завжди подаються стандартизовані дані, де центри очей та лінія рота знаходяться у фіксованих позиціях, незалежно від початкового положення голови користувача перед камерою [18].

Після виконання геометричного вирівнювання отримане зображення підлягає процедурі нормалізації та стандартизації. Це критичний етап підготовки даних, необхідний для забезпечення стабільної роботи глибоких нейронних мереж. Оскільки стандартні цифрові зображення кодуються цілими числами в діапазоні від 0 до 255 (формат RGB), пряме використання таких значень є неоптимальним. Великий розкид вхідних даних може призводити до нестабільності обчислювального процесу та сповільнювати навчання моделі.

Процес фінальної обробки складається з масштабування та нормалізації зображення. Спершу вирівняний зразок приводиться до фіксованого розміру, який відповідає вимогам вхідного шару моделі ArcFace. Наступним етапом виконується нормалізація, де значення яскравості пікселів переводяться у діапазон від -1 до 1. Ця операція реалізується через віднімання середнього значення та ділення на стандартне відхилення, що дозволяє центрувати дані та покращити збіжність нейромережі.

Результатом роботи алгоритму попередньої обробки є формування багатовимірного тензора, що містить інформацію про RGB канали та просторову структуру зображення. Саме цей стандартизований масив даних подається на вхід нейронної мережі для генерації унікального цифрового відбитка особи [11].

2.3 Математична модель верифікації та оцінка подібності

Результатом роботи згорткової нейронної мережі на етапі інференсу є генерація глибоких метричних ознак, які формують вектор фіксованої розмірності. Цей вектор являє собою компактний цифровий дескриптор, що

інкапсулює унікальні антропометричні характеристики особи в форматі, придатному для автоматизованої обробки [8].

Задача біометричної верифікації формалізується як процедура порівняння двох векторів:

- вектора запиту, який генерується системою в реальному часі на основі вхідного зображення з пристрою захоплення;
- еталонного вектора користувача, що був створений під час попередньої реєстрації та зберігається в базі даних [1, 5].

Для кількісної оцінки ступеня схожості між цими векторами необхідно визначити відповідну метрику відстані. У класичних задачах аналізу даних часто застосовується евклідова відстань, яка вимірює пряму лінійну дистанцію між точками у просторі. Однак у багатовимірних просторах, сформованих глибокими нейронними мережами, ефективність цієї метрики знижується [6]. Це зумовлено специфікою навчання моделі ArcFace: її функція втрат оптимізує розподіл ознак не у плоскому просторі, а на поверхні багатовимірної гіперсфери, використовуючи кутові відступи для розділення класів [12].

Виходячи з цієї геометричної особливості, найбільш природною та математично обґрунтованою метрикою для порівняння векторів у розробленій системі є косинусна подібність. Цей метод оцінює косинус кута між двома векторами, що дозволяє визначити їхню взаємну орієнтацію в просторі ознак незалежно від абсолютної величини (довжини) векторів, забезпечуючи високу точність класифікації при роботі з нормалізованими даними [13].

Косинусна подібність вимірює косинус кута між двома векторами. Її значення варіюється в діапазоні $[-1, 1]$, де:

- 1 означає, що вектори ідентичні (кут 0°);
- 0 означає, що вектори ортогональні;
- -1 означає, що вектори протилежні.

Математично ступінь подібності S розраховується як скалярний добуток векторів, поділений на добуток їхніх норм (2.5):

$$S(v_1, v_2) = \frac{v_1 \cdot v_2}{|v_1||v_2|} = \frac{\sum_{i=1}^d v_{1,i} v_{2,i}}{\sqrt{\sum_{i=1}^d v_{1,i}^2} \sqrt{\sum_{i=1}^d v_{2,i}^2}} \quad (2.5)$$

Важливою особливістю розроблюваної системи є те, що вихідні вектори нейронної мережі вже є нормалізованими (мають одиничну довжину, тобто $|v| = 1$). Це дозволяє суттєво спростити обчислення, звівши формулу лише до скалярного добутку (2.6):

$$S(v_1, v_2) = v_1 \cdot v_2 = \sum_{i=1}^d v_{1,i} \cdot v_{2,i} \quad (2.6)$$

Таке спрощення має велике практичне значення для оптимізації швидкодії системи, оскільки операція скалярного добутку виконується процесором значно швидше, ніж обчислення квадратних коренів [6].

Отримання числового значення косинусної подібності S є лише проміжним етапом. Для функціонування системи авторизації необхідно трансформувати це неперервне значення (від -1 до 1) у бінарне рішення «Доступ дозволено» або «Доступ заборонено». Ця процедура реалізується шляхом порівняння обчисленої подібності з фіксованим пороговим значенням T (threshold).

Формальне правило прийняття рішення виглядає наступним чином (2.7).

$$R(v_{\text{probe}}, v_{\text{gallery}}) = \begin{cases} \text{Match, якщо } S \geq T \\ \text{Non – Match, якщо } S < T \end{cases} \quad (2.7)$$

Визначення оптимального порогового значення класифікації є критичним етапом проектування, що встановлює баланс між рівнем захищеності та зручністю експлуатації системи. Оскільки імовірнісний характер біометричних алгоритмів унеможливорює повну відсутність похибок, налаштування чутливості завжди пов'язане з компромісом між помилками першого та другого роду. Зокрема, коефіцієнт помилкового допуску характеризує ризик несанкціонованої автентифікації сторонньої особи, що становить пряму загрозу безпеці. Водночас коефіцієнт помилкової відмови відображає ймовірність невизнання легітимного користувача, що знижує пропускну здатність системи та зручність її використання [1].

Математично ці ймовірності залежать від обраного порогу T і розподілу ймовірностей для «своїх» та «чужих» (2.8):

$$\begin{aligned} \text{FAR}(T) &= P(S \geq T \mid \text{impostor}) \\ \text{FRR}(T) &= P(S < T \mid \text{genuine}) \end{aligned} \quad (2.8)$$

Процедура прийняття рішення про авторизацію базується на порівнянні обчисленого коефіцієнта косинусної подібності із заздалегідь визначеним граничним значенням. Якщо ступінь схожості векторів перевищує цей ліміт, система класифікує пред'явлене обличчя як таке, що належить власнику облікового запису. Для розроблюваної системи, враховуючи вимоги до безпеки корпоративних ресурсів, пропонується встановити поріг, що відповідає значенню ймовірності помилкового допуску 0,1% (одна помилка допуску на 1000 спроб злому). Такий рівень захищеності узгоджується з рекомендаціями міжнародних стандартів цифрової ідентифікації та дозволяє мінімізувати ризики успішних атак на підміну особи [5].

Конкретне числове значення порога визначається емпіричним шляхом на етапі калібрування системи шляхом аналізу результатів тестування на валідаційній вибірці. Процес налаштування передбачає пошук балансу між двома конфліктуючими показниками: ймовірністю помилкового допуску сторонньої особи та ймовірністю помилкової відмови легітимному користувачеві [1]. У рамках даного дослідження пріоритет надається мінімізації першого показника, оскільки забезпечення цілісності периметра безпеки є критичнішим завданням, ніж незначне зниження зручності користування, яке може виникнути через необхідність повторного сканування обличчя.

Висновки до розділу 2

У другому розділі здійснено математичне моделювання та алгоритмічне проектування ключових компонентів системи біометричної верифікації. Зокрема, обґрунтовано доцільність використання глибокої згорткової нейронної мережі архітектури ArcFace, яка завдяки специфічній функції втрат забезпечує

формування високодискримінативних ознак та компактну кластеризацію векторів на гіперсфері.

Для нівелювання впливу неконтрольованих умов зйомки та підвищення точності розпізнавання розроблено алгоритм попередньої обробки даних на базі каскадних мереж MTCNN. Реалізація етапів детекції, вирівнювання за ключовими точками та нормалізації яскравості дозволяє привести вхідні зображення до канонічного вигляду, необхідного для коректної роботи нейромережі.

Крім того, формалізовано математичну процедуру прийняття рішення про верифікацію, що базується на розрахунку метрики косинусної подібності. Також визначено критерії оцінки надійності системи через співвідношення помилок першого (FAR) та другого (FRR) роду. Запропоновані математичні моделі та алгоритмічні рішення складають необхідний базис для програмної реалізації та експериментального дослідження системи, що є предметом розгляду у наступному розділі роботи.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

3.1 Обґрунтування вибору програмних та апаратних засобів реалізації

Проектування архітектури програмного комплексу для біометричної ідентифікації вимагає системного підходу до вибору інструментальних засобів, оскільки ефективність кінцевого продукту визначається не лише якістю алгоритмів, а й середовищем їх виконання. Специфіка поставленої задачі полягає в необхідності органічного поєднання двох різнорідних обчислювальних процесів: виконання ресурсномістких математичних операцій над багатовимірними масивами даних (тензорами) та забезпечення високошвидкісної асинхронної обробки мережових запитів у реальному часі.

Виходячи з цих передумов, формування технологічного стека здійснювалося на основі критеріїв швидкодії, масштабованості та сумісності компонентів. Пріоритет надавався рішенням із відкритим вихідним кодом, що мають активну підтримку професійної спільноти та розвинену екосистему спеціалізованих бібліотек. У даному підрозділі обґрунтовано вибір базової мови програмування, веб-фреймворку, системи управління базами даних та середовища розгортання, сукупність яких забезпечує баланс між продуктивністю обчислень, надійністю зберігання даних та швидкістю розробки [19, 20].

В якості основної мови реалізації серверної частини обрано Python версії 3.11. Цей вибір є стратегічним, оскільки даний реліз містить результати багаторічного проекту «Faster CPython». Згідно з офіційною технічною документацією Python Software Foundation, версія 3.11 демонструє приріст продуктивності в середньому на 25% порівняно з версією 3.10 завдяки впровадженню адаптивного інтерпретатора байт-коду [19]. Для задач машинного навчання та вебсервісів це дозволяє знизити затримки без необхідності переписувати код на компільовані мови.

Для забезпечення надійності середовища розробки використано менеджер залежностей Poetry. На відміну від класичних інструментів, Poetry використовує

файл конфігурації `pyproject.toml` та файл блокування `poetry.lock`. Це гарантує сувору фіксацію версій усіх бібліотек, що є критичним для стабільності роботи системи при перенесенні коду з локального комп'ютера на сервер.

Ядром серверного додатку виступає фреймворк FastAPI. Вибір даного інструменту обумовлений його архітектурою, побудованою на стандарті ASGI, та використанням бібліотек `anyio` і `starlette`, що дозволяє серверу обробляти запити в асинхронному режимі за допомогою синтаксису `async/await`. Це архітектурне рішення є критично важливим для реалізованого алгоритму біометричної верифікації, який являє собою складний багатоступеневий процес. Обробка кожного запиту включає послідовність ресурсоємних операцій: зчитування вхідного бінарного потоку безпосередньо в оперативну пам'ять, декодування зображення у матрицю пікселів засобами бібліотеки OpenCV, виконання математичних операцій згортки для отримання векторних ознак моделлю ArcFace та, нарешті, виконання пошукового запиту до бази даних PostgreSQL (Рис. 3.1).

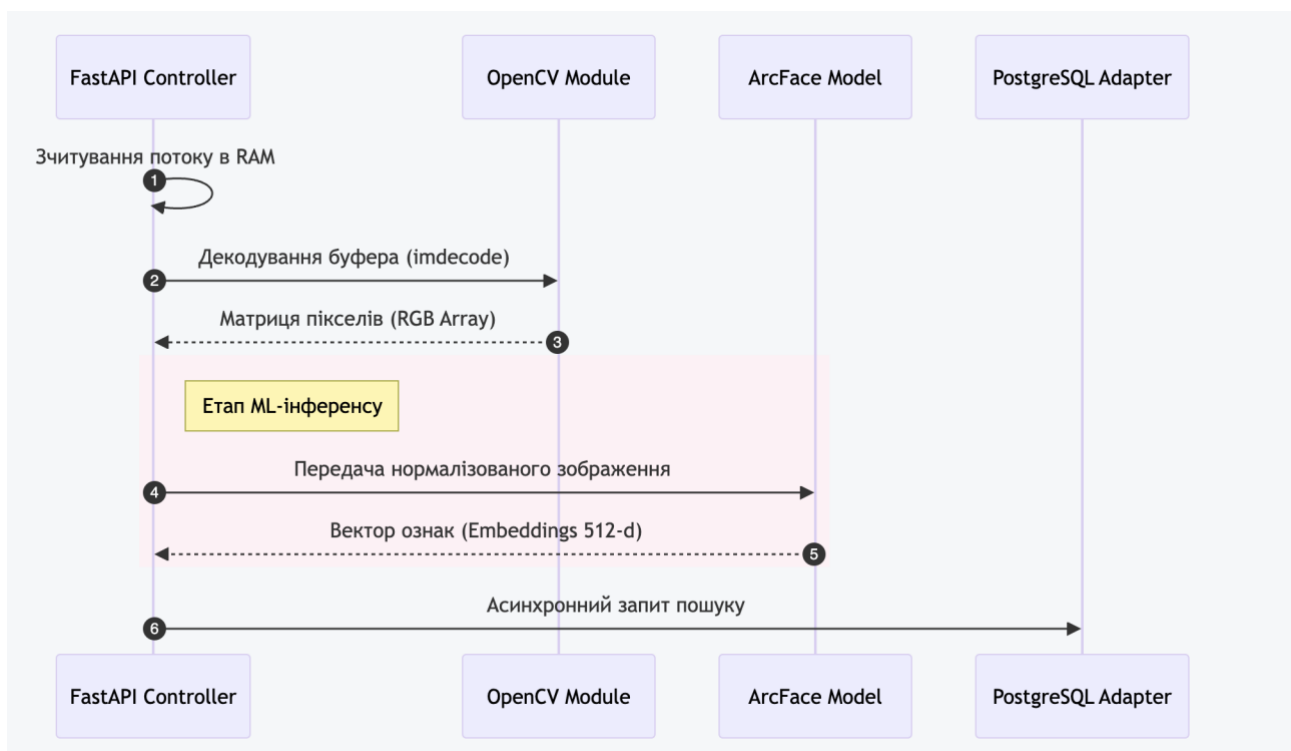


Рис. 3.1 – Діаграма послідовності генерації біометричного шаблону на сервері

Асинхронна модель дозволяє ефективно розподіляти системні ресурси: у моменти очікування відповіді від підсистеми вводу-виводу, зокрема при

зверненні до бази даних, потік виконання не блокується, а перемикається на обробку нових вхідних з'єднань або завершення обчислень для інших користувачів. Такий підхід забезпечує максимальну утилізацію процесорного часу та високу пропускну здатність системи навіть при значному навантаженні.

Окремої уваги заслуговує програмна реалізація конвеєра обробки зображень та передачі даних. У розробленій системі відмовлено від класичного підходу, що передбачає проміжне збереження завантажених файлів на жорсткий диск сервера, оскільки операції вводу/виводу є найповільнішим компонентом у ланцюгу обробки запиту. Натомість реалізовано механізм обробки даних виключно в оперативній пам'яті. Вхідний файл, отриманий через REST API, зчитується у бінарний буфер, трансформується у масив цілих чисел без знаку засобами бібліотеки NumPy та декодується бібліотекою OpenCV. Такий підхід дозволяє мінімізувати затримки та уникає накопичення тимчасових файлів, що позитивно впливає на загальну швидкодію системи.

Ключовим компонентом продуктивності API є бібліотека валідації даних Pydantic. Використана у проекті друга версія Pydantic базується на ядрі pydantic-core, написаному на системній мові Rust. Це забезпечує високу швидкість серіалізації та десеріалізації JSON-структур, перевершуючи чисті Python-аналоги у 5-10 разів [21].

Для розгортання серверної частини вебдодатку реалізовано стандартну для Python-середовища архітектуру, що поєднує можливості Uvicorn та Gunicorn. Зокрема, Uvicorn використовується як високопродуктивний ASGI-сервер, що відповідає за безпосередній прийом мережових з'єднань та виконання асинхронного коду. Управління цим процесом покладено на WSGI HTTP-сервер Gunicorn, який виступає в ролі менеджера процесів. Він керує пулом робочих екземплярів Uvicorn, що забезпечує ефективне розпаралелювання навантаження між всіма доступними ядрами процесора та гарантує відмовостійкість системи завдяки механізму автоматичного перезапуску процесів у разі виникнення критичних збоїв.

Архітектура клієнтської частини побудована за подієво-орієнтованою моделлю. Взаємодія з обладнанням користувача реалізована через WebRTC API, зокрема інтерфейс `MediaDevices.getUserMedia`, який забезпечує трансляцію відеопотоку з веб-камери на HTML-елемент відео у реальному часі. Важливою особливістю реалізації є те, що відеопотік використовується виключно для візуального контролю користувачем свого положення в кадрі і не передається на сервер у фоновому режимі. Це забезпечує дотримання принципів конфіденційності та економії мережевого трафіку.

Такий підхід дозволяє мінімізувати затримки та уникає накопичення тимчасових файлів на сервері. Для отримання біометричних векторів використано метод `DeerFace.represent` [23]. Важливим архітектурним рішенням є активація параметра суворої детекції. Це діє як фільтр першого рівня: якщо алгоритм MTCNN не може з високою ймовірністю локалізувати обличчя на кадрі (наприклад, зображення надто розмите, темне або містить сторонні предмети), система генерує виключення `HTTPException 400`. Це запобігає подачі «сміттєвих» даних на вхід нейромережі `ArcFace` та гарантує, що у базі даних зберігаються лише валідні еталони.

Фінальним етапом процесу (Рис. 3.2) є прийняття рішення про авторизацію на основі даних, отриманих від системи керування базами даних. Серверна логіка ініціює виконання SQL-запиту до таблиці користувачів, використовуючи оператор косинусної відстані. Завдяки використанню індексу HNSW, база даних повертає єдиний запис, вектор якого є геометрично найближчим до вектора запиту, разом із числовим значенням цієї відстані.

Отримане значення дистанції підлягає перевірці умовою розгалуження. Якщо дистанція перевищує встановлений пороговий критерій (0.68), система класифікує спробу як невдалу. У цьому випадку сервер перериває обробку та повертає клієнту HTTP-відповідь зі статусом `401 Unauthorized`, що інтерпретується інтерфейсом як помилка ідентифікації.

У випадку, якщо дистанція менша або дорівнює порогу, система переходить до процедури створення сесії. Модуль автентифікації генерує токен

доступу, у корисне навантаження якого записується ідентифікатор користувача та його роль. Токен підписується секретним ключем сервера за алгоритмом HS256 для запобігання підробці. Сформований токен повертається клієнту у тілі відповіді зі статусом 200 OK. Отримавши підтвердження, клієнтський додаток зберігає токен у локальному сховищі та виконує перенаправлення користувача до захищеного розділу системи.

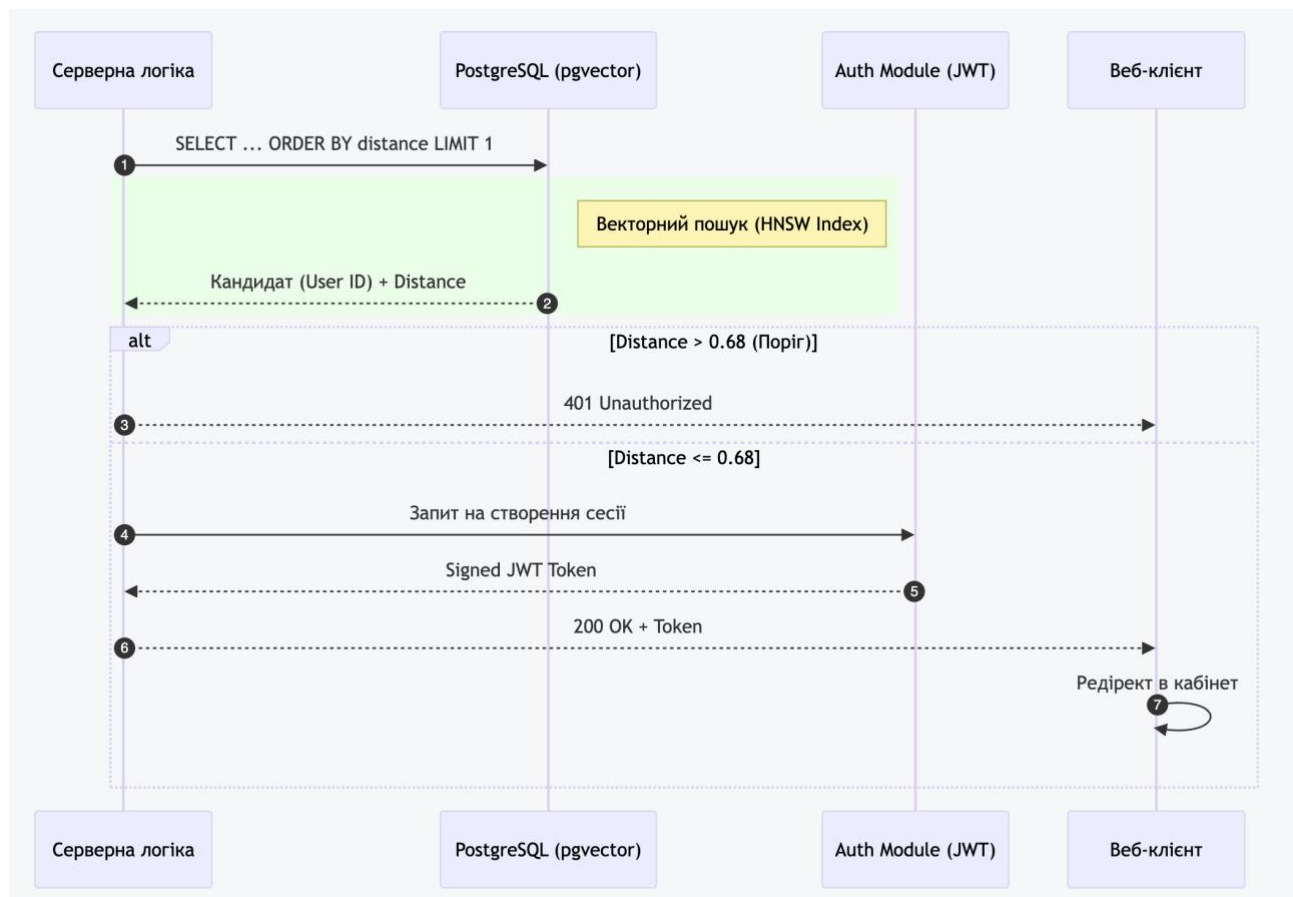


Рис. 3.2 – Алгоритм верифікації користувача та генерації токена доступу

Це значення було обрано на основі рекомендацій для моделі ArcFace при використанні метрики косинусної відстані. Варто зазначити, що у бібліотеці DeerFace та базі даних використовується саме відстань, яка пов'язана з подібністю співвідношенням:

$$\text{Distance} = 1 - \text{Similarity} \quad (3.1)$$

Пошук ідентичності реалізовано за алгоритмом 1:N (One-to-Many). Замість завантаження всіх векторів з бази даних у пам'ять додатку та їх перебору циклом Python, задача пошуку найближчого сусіда делегується СУБД PostgreSQL. Запит

повертає лише одного кандидата (LIMIT 1) з мінімальною відстанню, після чого додаток перевіряє, чи вписується ця відстань у допустимий поріг T.

3.2 Експериментальне дослідження ефективності архітектур розпізнавання

Вибір нейронної мережі для біометричної системи є складною задачею оптимізації, що вимагає пошуку компромісу між точністю верифікації та споживанням апаратних ресурсів. Для об'єктивного вибору архітектури було розроблено автоматизований програмний модуль мовою Python.

Для об'єктивного вибору архітектури було розроблено автоматизований програмний комплекс на мові Python, який виконує порівняльне тестування трьох найпоширеніших моделей: VGG-Face, FaceNet та ArcFace.

Для проведення експерименту було обрано набір даних LFW [24]. Цей датасет є стандартом для таких задач, оскільки містить зображення людей у неконтрольованих умовах, а не в студійних. Це дозволяє перевірити стійкість системи до реальних факторів – зміна освітлення, повороти голови чи різні емоції, що максимально наближає умови тестування до реальної експлуатації.

Реалізований у програмному скрипті алгоритм тестування передбачає комплексну перевірку ефективності системи, що розпочинається з етапу підготовки даних. За допомогою спеціалізованої функції формується збалансована тестова вибірка, яка включає як позитивні пари, так і негативні, що є необхідною умовою для коректного розрахунку метрик бінарної класифікації.

На першому етапі відбувається ініціалізація структур даних (Рис. 3.3) для накопичення результатів експерименту. Створюються списки для зберігання істинних міток (`y_true`), бінарних передбачень системи (`y_pred_labels`) та числових оцінок впевненості (`y_scores`), які необхідні для подальшої побудови ROC-кривих.

```
def test_model(model, pairs): 1 usage  Ⓐ Bohdan Babytskyi
    results = {}

    y_true = []
    y_pred_labels = []
    y_scores = []

    total_time = 0
    peak_mem = 0
```

Рис. 3.3 – Ініціалізація структур даних

Найважливішою частиною алгоритму є цикл проходу по тестових парах (Рис. 3.4), в якому виконується безпосереднє вимірювання витрат ресурсів. Вимірювання часу виконання здійснюється для кожного окремого запиту, що дозволяє отримати середній час інференсу без урахування можливих мережових затримок. Для оцінки споживання оперативної пам'яті використано бібліотеку `tracemalloc`. Вона дозволяє зафіксувати пікове використання пам'яті саме в момент виконання найбільш ресурсномістких математичних операцій згортки.

```
for img1, img2, is_same in pairs:
    start_time = time.time()
    tracemalloc.start()

    try:
        result = DeepFace.verify(img1_path=img1, img2_path=img2, model_name=model, enforce_detection=False)
        pred_same = result["verified"]
        distance = result["distance"]

        score = -distance

        y_true.append(1 if is_same else 0)
        y_pred_labels.append(1 if pred_same else 0)
        y_scores.append(score)

    except Exception as e:
        print(f"Помилка для {model}: {e}")

    end_time = time.time()
    current, peak = tracemalloc.get_traced_memory()
    tracemalloc.stop()

    total_time += end_time - start_time
    peak_mem = max(peak_mem, peak)
```

Рис. 3.4 – Цикл проходу по тестових парах та вимірювання витрат ресурсів.

Завершальним етапом є кількісна оцінка надійності моделей з використанням інструментарію бібліотеки `sklearn.metrics`. Вона включає розрахунок загальної точності класифікації та площі під кривою помилок (ROC AUC), що характеризує здатність моделі ефективно розрізняти класи незалежно від обраного порогового значення прийняття рішень.

На основі проведених експериментальних досліджень, результати яких відображені на ROC-кривих (Рис. 3.5) та у зведеній таблиці метрик (Таблиця 3.1), було виконано комплексне порівняння ефективності трьох архітектур нейронних мереж: VGG-Face, Facenet та ArcFace. Аналіз якості розпізнавання продемонстрував, що беззаперечним лідером є модель VGG-Face, яка досягла найвищих показників точності (Accuracy 0.92) та площі під кривою (AUC 0.98). Графічна інтерпретація результатів підтверджує її стабільність, показуючи найкраще співвідношення чутливості до хибних спрацьовувань. Конкурентну позицію зайняла модель ArcFace, яка при ідентичній точності (0.92) показала лише незначне відставання за показником AUC (0.95), тоді як Facenet продемонструвала найнижчу ефективність серед протестованих алгоритмів (Accuracy 0.87).

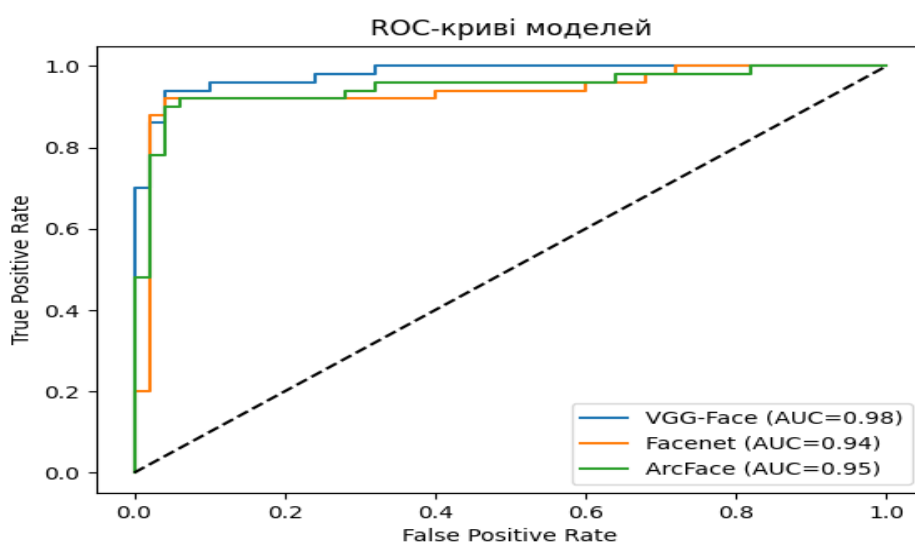


Рис. 3.5 – Порівняння ROC-кривих та значень AUC для моделей VGG-Face, Facenet та ArcFace

Розглядаючи аспект обчислювальної ефективності (Рис. 3.6), спостерігається чіткий компроміс між швидкістю обробки та споживанням

ресурсів. VGG-Face, попри свою масивність та найбільше споживання пікової пам'яті (1109.53 МБ), виявилася найшвидшою моделлю із середнім часом перевірки 0.3821 с. На противагу їй, архітектура Facenet є найбільш економною до ресурсів (144.15 МБ), проте її швидкодія (1.0888 с) є найнижчою, що може бути критичним для систем реального часу. Модель ArcFace виступає збалансованим рішенням, демонструючи оптимальне співвідношення високої швидкості (0.4791 с) та помірному споживанню пам'яті (258.42 МБ).

Таблиця 3.1

Результати експериментів

	VGG-Face	Facenet	ArcFace
Ассурасу	0.92	0.87	0.92
AUC	0.98	0.94	0.95
Середній час на перевірку	0.3821 сек	1.0888 сек	0.4791 сек
Пікова пам'ять	1109.53 МБ	144.15 МБ	258.42 МБ

Результати тестів показали, що різниця в точності між групами мінімальна (менше 0.08). Це підтверджує, що функція втрат Additive Angular Margin, закладена в ArcFace, працює коректно: вона примусово "стягує" вектори одного класу на гіперсфері, незалежно від кольору шкіри чи статі. Для нас це означає, що систему можна впроваджувати в реальних умовах без ризику отримати скарги на дискримінацію або некоректну роботу для певних категорій користувачів.

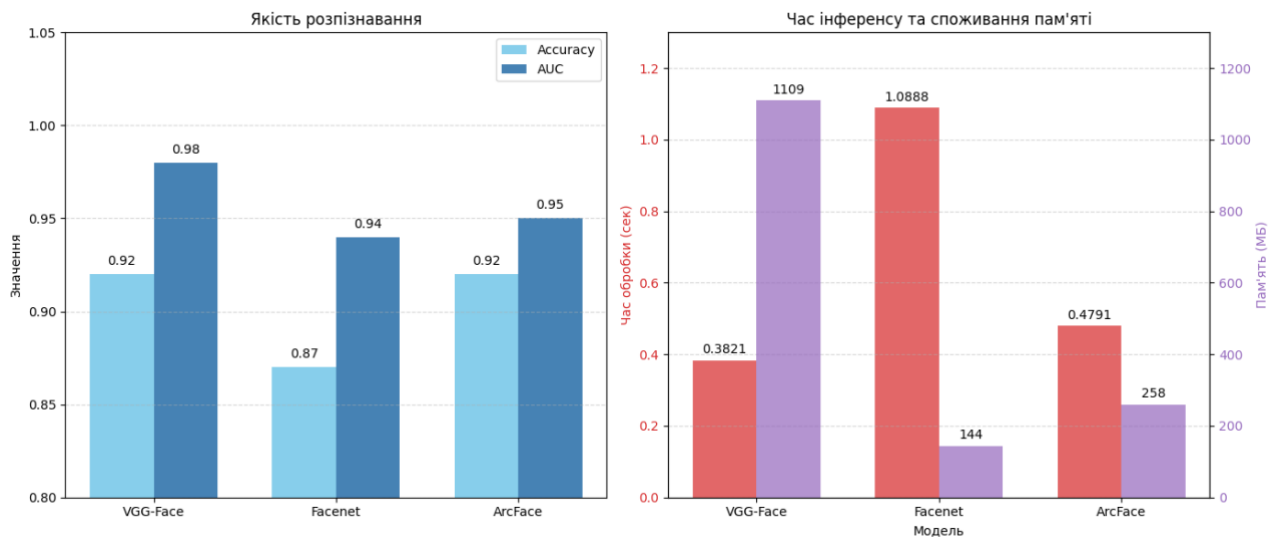


Рис. 3.6 – Оцінка ефективності нейронних мереж за критеріями швидкодії, споживання пам'яті та якості класифікації

Таким чином, вибір оптимальної архітектури залежить від специфіки апаратного забезпечення цільової платформи. Для реалізації системи на потужних серверних станціях, де обсяг оперативної пам'яті не є лімітуючим фактором, рекомендовано використовувати VGG-Face, яка забезпечує миттєву реакцію та максимальну валідність результату. Водночас для умов з обмеженими апаратними ресурсами, хмарних середовищ або вбудованих систем доцільнішим вибором є ArcFace, що гарантує високу точність ідентифікації при оптимізованому використанні пам'яті.

3.3 Дослідження стійкості системи до демографічних факторів та аналіз алгоритмічної упередженості

Впровадження біометричних систем ідентифікації у публічний простір вимагає ретельної перевірки на відповідність етичним нормам штучного інтелекту. Важливим аспектом, який необхідно враховувати при розробці, є схильність нейромереж до алгоритмічної упередженості. На практиці це явище проявляється у нерівномірній точності роботи алгоритму: статистика помилок часто суттєво зростає при обробці зображень представників певних демографічних груп, що знижує загальну надійність системи [3].

Для оцінки надійності обраної моделі ArcFace до демографічних варіацій було розроблено спеціалізований програмний модуль BiasAnalyzer, код якого наведено у Додатку А.

Традиційні датасети для навчання нейромереж часто мають суттєвий дисбаланс у бік європеоїдної раси. Для проведення об'єктивного експерименту було обрано датасет FairFace, який створено з метою мінімізації упередженості. Він містить збалансовану кількість зображень, розподілених за 7 расовими групами: White, Black, Latino/Hispanic, East Asian, Southeast Asian, Indian, Middle Eastern.

В основу роботи аналізатора покладено оцінку щільності кластеризації векторів ознак усередині кожної демографічної групи. Алгоритм розраховує середню косинусну відстань між усіма парами зображень однієї особи в межах групи. Чим менше це значення, тим компактніше неймережа групує представників даної категорії, і тим вищою є ймовірність правильної верифікації.

Скрипт виконав генерацію ембедінгів та розрахунок метрик для вибірки з 500 зображень. Результати оцінки середньої дистанції для різних расових груп наведено на рис 3.7.

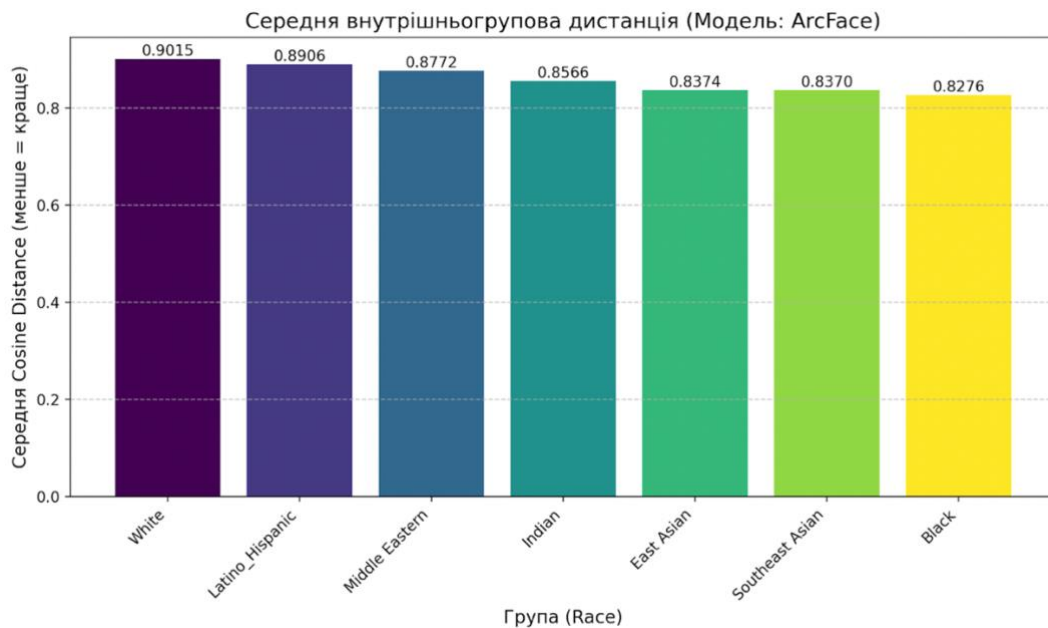


Рис. 3.7 – Гістограма середньої внутрішньогрупової відстані за расовою

Аналогічну методологію було застосовано для дослідження потенційної гендерної упередженості алгоритму. На рисунку 3.8 наведено порівняльний

аналіз середньої внутрішньогрупової дистанції для чоловічої та жіночої вибірок, візуалізований засобами бібліотеки Matplotlib [25].

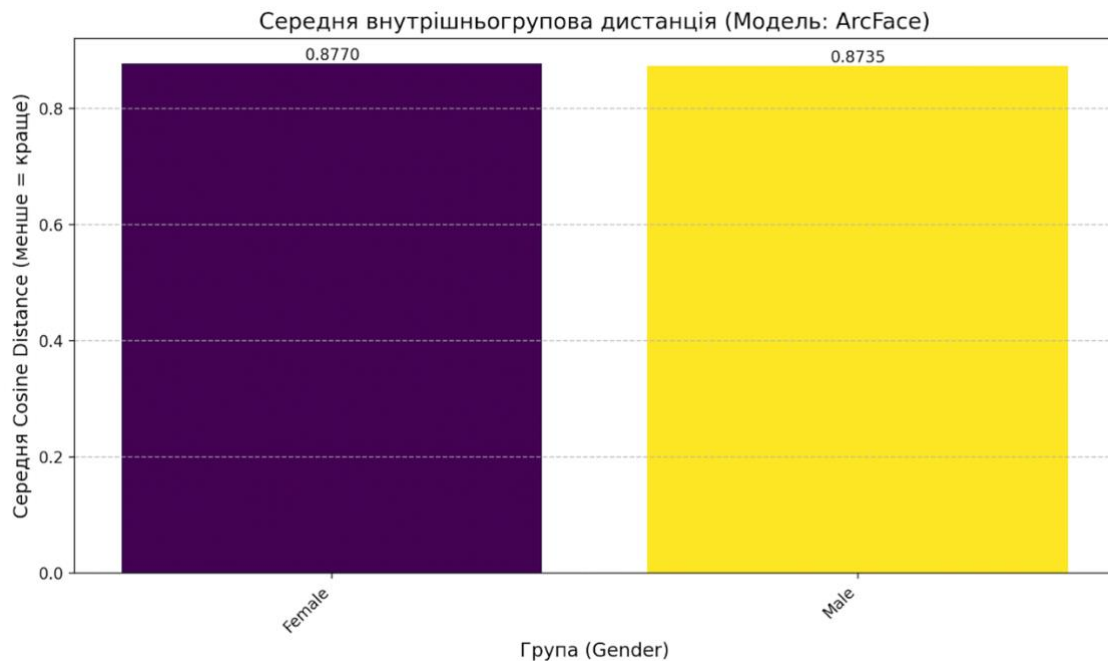


Рис. 3.8 – Порівняння середньої дистанції за гендерною ознакою

Різниця між чоловічою та жіночою вибірками є мінімальною (близько 0.035), що підтверджує відсутність суттєвого гендерного дисбалансу в архітектурі ArcFace.

Результати тестів показали, що різниця в точності між групами мінімальна (менше 0.08). Це підтверджує, що функція втрат Additive Angular Margin, закладена в ArcFace, працює коректно: вона примусово "стягує" вектори одного класу на гіперсфері, незалежно від кольору шкіри чи статі. Для нас це означає, що систему можна впроваджувати в реальних умовах без ризику отримати скарги на дискримінацію або некоректну роботу для певних категорій користувачів.

3.4 Проєктування архітектури клієнтського додатку та інтерфейсу користувача

Розроблений програмний комплекс реалізовано як монолітний вебсервіс, побудований за патерном MVC, адаптованим для асинхронного стеку Python. Роль контролера виконує фреймворк FastAPI, моделлю виступає структура бази

даних PostgreSQL, а представленням – HTML-шаблони, що рендеряться на стороні сервера за допомогою рушія Jinja2.

Основною в базі даних є таблиця `users`. Її структуру спроектовано так, щоб зберігати в межах одного запису як стандартну текстову інформацію, так і біометричний вектор обличчя. Такий підхід дозволяє об'єднати класичну авторизацію з біометричною без необхідності розносити дані по різних сховищах.

Як видно з Рис. 3.9, у структурі таблиці, окрім зберігання біометричного вектора типу `vector(512)`, передбачено поле `role`. Це дозволяє реалізувати рольову модель доступу. Також використано індекс HNSW [27], який оптимізовано для косинусної метрики. Це дозволяє виконувати пошук найближчого сусіда з логарифмічною складністю, забезпечуючи миттєвий відгук системи.

```
CREATE EXTENSION IF NOT EXISTS vector;  
  
DROP TABLE IF EXISTS users;  
  
CREATE TABLE users (  
    id bigserial PRIMARY KEY,  
    email text NOT NULL UNIQUE,  
    password_hash text NOT NULL,  
    face_embedding vector(512),  
    role VARCHAR(50) NOT NULL DEFAULT 'user',  
    created_at timestamptz DEFAULT now()  
);  
  
CREATE INDEX ON users USING hnsw (face_embedding vector_cosine_ops);
```

Рис. 3.9 – Схема бази даних з підтримкою векторного індексу HNSW

Архітектура шару даних базується на розширенні функціональних можливостей СУБД PostgreSQL шляхом активації модуля `pgvector`. Це дозволяє використовувати спеціалізований тип даних `vector(512)` для безпосереднього зберігання у реляційній таблиці 512-вимірних біометричних ембедінгів, згенерованих нейромережею ArcFace. Такий підхід забезпечує консолідацію

даних у єдиному сховищі, виключаючи необхідність залучення зовнішніх NoSQL-рішень або файлових систем для зберігання векторів. Високопродуктивна взаємодія з базою даних реалізована через асинхронний драйвер `asynpcrg` із використанням пулу з'єднань. Для забезпечення прозорості інтеграції типів даних на етапі ініціалізації додатку зареєстровано кастомний кодек, який виконує автоматичну серіалізацію масивів `numpy.array` у внутрішній формат PostgreSQL, що суттєво спрощує процедуру об'єктно-реляційного відображення.

У системі реалізовано суворі заходи безпеки щодо зберігання автентифікаційних даних. Відповідно до сучасних стандартів кібербезпеки, зокрема рекомендацій NIST SP 800-63-3 [5], зберігання паролів у відкритому вигляді суворо заборонено, а для їх захисту необхідно використовувати криптографічно стійкі алгоритми хешування з використанням солі для захисту від атак за допомогою райдужних таблиць.

Для захисту облікових даних користувачів у розробленій системі використано спеціалізовану криптографічну бібліотеку `passlib` із застосуванням алгоритму `bcrypt`. На відміну від звичайних криптографічних хеш-функцій, `bcrypt` є адаптивним алгоритмом. Це означає, що він спеціально спроектований бути повільним при обчисленні, що робить атаки методом повного перебору та використання райдужних таблиць економічно не вигідними для злоумисника.

Крім того, `bcrypt` автоматично генерує та зберігає унікальну сіль (`salt`) для кожного хешу всередині самого рядка результату, що позбавляє необхідності окремо керувати стовпцем солі в базі даних.

Реалізація модуля хешування наведена на Рис. 3.10.

```

from passlib.context import CryptContext

pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")

def verify_password(plain_password, hashed_password): 1 usage new *
    return pwd_context.verify(plain_password, hashed_password)

def get_password_hash(password): 1 usage new *
    return pwd_context.hash(password)

```

Рис. 3.10 – Реалізація хешування та верифікації паролів за допомогою Passlib

Завдяки використанню CryptContext, система отримує гнучкість: у разі необхідності зміни алгоритму хешування в майбутньому, достатньо буде лише змінити конфігурацію у параметрі schemes, не переписуючи логіку перевірки по всьому додатку.

У базі даних у полі password_hash зберігається результуючий рядок, який містить інформацію про алгоритм, вартість обчислення, сіль та власне хеш. Це забезпечує надійну верифікацію користувача при вході в систему.

API Ендпоінт реєстрації POST /register приймає форму (multipart/form-data), що містить email, пароль та файл зображення. Важливою особливістю реалізації є перевірка на унікальність користувача перед записом.

Найбільш технологічно складним компонентом системи є модуль входу (/login), який підтримує мультимодальну автентифікацію. Ключова інновація полягає у перенесенні обчислювального навантаження з порівняння векторів з рівня додатку на рівень бази даних.

У разі обрання опції type="biometric" система ініціює алгоритм верифікації, що розпочинається з проходження вхідного зображення через каскад детектора та згорткової нейромережі для генерації унікального пошукового вектора. Сформований векторний опис безпосередньо використовується як аргумент у запиті до бази даних, яка виконує операцію

пошуку «найближчого сусіда» для визначення найбільш релевантного еталона серед масиву збережених записів.

Замість завантаження всіх векторів користувачів у пам'ять Python і їх перебору у циклі (що мало б лінійну складність $O(N)$ і було б неефективним при зростанні бази), використано оптимізований SQL-запит із застосуванням оператора косинусної відстані, наведена на Рис. 3.11.

```
SELECT face_embedding <=> $1 AS distance
FROM users
ORDER BY distance
LIMIT 1;
```

Рис. 3.11 – Фрагмент виконання запиту біометричної верифікації на рівні бази даних

Оператор `<=>` (cosine distance operator), наданий розширенням `pgvector`, обчислює ступінь відмінності між вектором запиту (1) та збереженими векторами. Конструкція `ORDER BY ... LIMIT 1` гарантує, що база даних поверне лише одного кандидата з мінімальною відстанню. Завдяки підтримці індексів IVFFlat (Inverted File with Flat Compression), цей пошук може виконуватися з логарифмічною складністю, що забезпечує миттєвий відгук навіть при мільйонах записів.

Після отримання кандидата система виконує фінальну перевірку на стороні серверу. Поріг $T = 0.68$ слугує гарантом безпеки: якщо навіть база даних знайшла "найбільш схожу" людину, але ступінь схожості недостатній, відповідно доступ буде відхилено.

У разі успішної автентифікації (незалежно від обраного методу: за паролем чи біометрією) система повинна підтримувати сеанс користувача. Для цього замість класичних серверних сесій використано стандарт JWT [28], що дозволяє реалізувати архітектуру без збереження стану.

Механізм працює наступним чином, зокрема при успішному вході сервер генерує токен доступу, який криптографічно підписується секретним ключем за алгоритмом HS256. У корисне навантаження токена інкапсулюється ідентифікатор користувача та його роль, це дозволяє серверу в подальшому

валідувати права доступу та ідентифікувати клієнта при кожному наступному запиті без необхідності повторного звернення до бази даних, що суттєво знижує навантаження на систему. З міркувань безпеки час життя токена обмежено 30 хвилинами, що мінімізує ризики у випадку його перехоплення.

Для забезпечення взаємодії з кінцевим користувачем розроблено вебінтерфейс, побудований за технологією Server-Side Rendering (SSR). Використання шаблонізатора Jinja2 дозволяє динамічно генерувати HTML-сторінки на стороні сервера, підставляючи дані користувача у заздалегідь підготовлені шаблони.

Архітектурна організація клієнтської частини базується на використанні динамічних HTML-шаблонів, що забезпечують інтерфейс взаємодії користувача з системою. Зокрема, шаблони `templates/register.html` та `templates/login.html` містять інтерактивні форми для реєстрації та автентифікації, функціонал яких розширено засобами JavaScript. Використання технології Fetch API дозволяє реалізувати асинхронну передачу біометричних даних (зображень) на сервер без необхідності повного перезавантаження сторінки, що забезпечує високу чутливість інтерфейсу. У разі успішної авторизації користувач перенаправляється до захищеної панелі управління (`templates/cabinet.html`), де візуалізуються його унікальний ідентифікатор та поточний статус у системі.

Важливою функцією, реалізованою в системі, є можливість видалення облікового запису. Цей функціонал є не лише технічною опцією, але й вимогою регламенту GDPR [29], зокрема статті 17 «Право на забуття» та національного законодавства [30, 29].

Оскільки біометричні дані (вектори обличчя) відносяться до категорії чутливих персональних даних, система повинна гарантувати їх повне знищення за запитом власника. У коді це реалізовано через прямий SQL-запит (Рис. 3.12).

```

@app.post("/delete_account")  ⚡ Bohdan Babytskyi
async def delete_account(current_user: dict = Depends(get_current_user)):
    user_id_to_delete = current_user['id']
    async with DB_POOL.acquire() as connection:
        result = await connection.execute("DELETE FROM users WHERE id = $1", user_id_to_delete)
        if result == "DELETE 1":
            return JSONResponse(content={"status": "success", "message": "Акаунт успішно видалено."})
        else:
            raise HTTPException(status_code=404, detail="Користувача не знайдено.")

```

Рис. 3.12 – Запит на видалення акаунту

Це гарантує, що разом із записом користувача безповоротно видаляється і його біометричний зліпок, що унеможливорює подальше використання цих даних системою.

Для організації взаємодії користувача з програмним комплексом розроблено вебінтерфейс на базі шаблонізатора Jinja2, який забезпечує динамічну генерацію сторінок безпосередньо на сервері. На рисунку 3.13 наведено головну сторінку авторизації. Її дизайн виконано в мінімалістичному стилі, що дозволяє уникнути візуального шуму та сфокусувати увагу користувача виключно на процесі біометричної ідентифікації. Верстка сторінки є адаптивною, що забезпечує коректне відображення інтерфейсу як на стаціонарних моніторах, так і на мобільних пристроях.

Інтерфейс реалізує гібридну модель доступу, надаючи пріоритет безконтактній взаємодії. За замовчуванням активується режим входу через веб-камеру, проте передбачено і резервний механізм авторизації за паролем (fallback mechanism). Наявність альтернативного сценарію є критично важливою для забезпечення відмовостійкості системи: це гарантує доступ до акаунту навіть у випадках апаратних збоїв камери, відсутності драйверів або умов недостатнього освітлення, коли оптичне розпізнавання є неможливим.

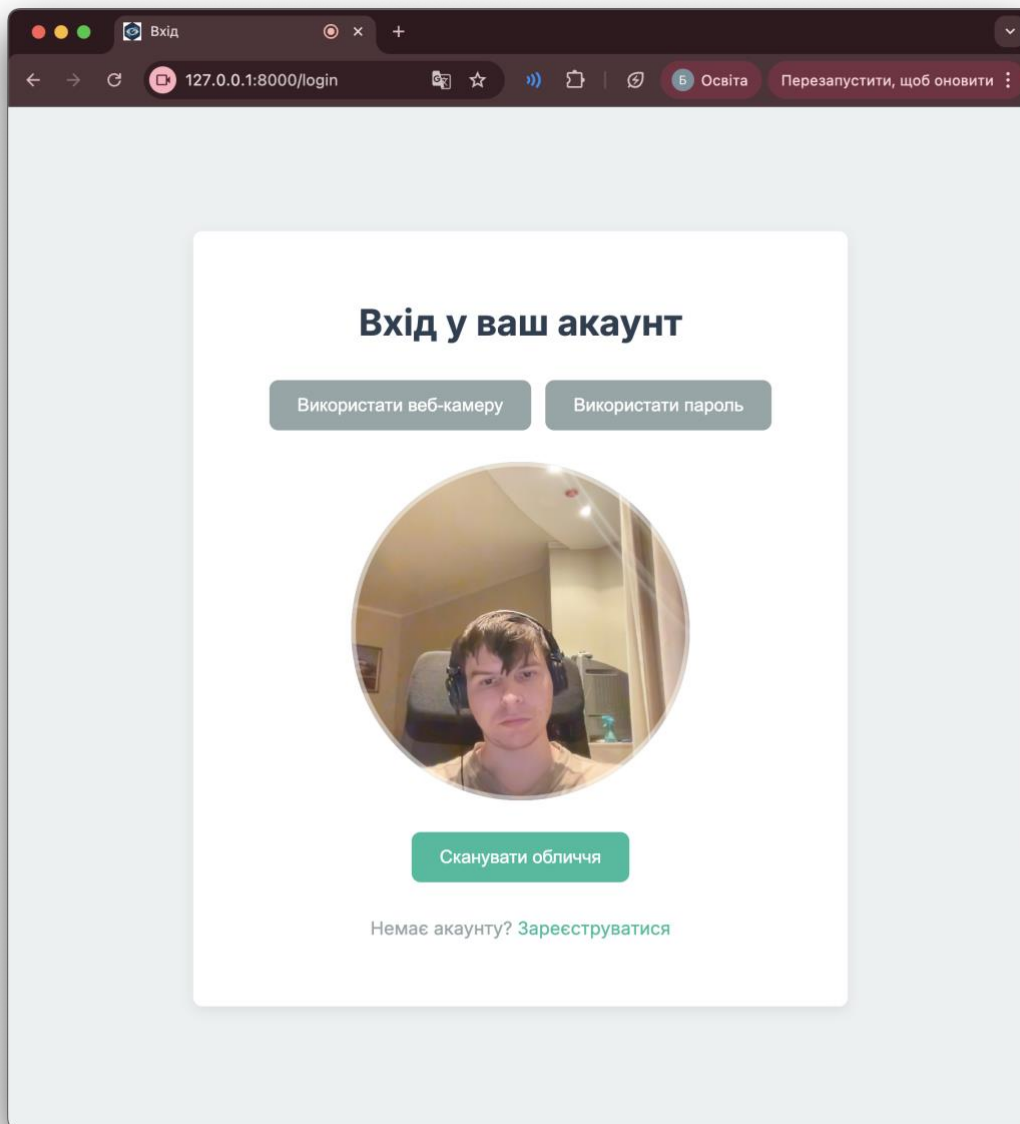


Рис. 3.13 – Головна сторінка системи біометричної ідентифікації

Центральним елементом візуальної композиції є область попереднього перегляду відеопотоку. Вона реалізована з використанням HTML5 Video API та дозволяє користувачеві в реальному часі контролювати своє положення в кадрі, уникаючи ситуацій, коли обличчя частково перекрите або знаходиться поза фокусом.

Запропонована архітектура взаємодії клієнт-сервер побудована на принципах RESTful API, що забезпечує чітку стандартизацію обміну даними у форматі JSON. Такий підхід робить систему масштабованою: у перспективі це

дозволяє розробити нативні мобільні додатки або інтегрувати модуль біометрії в сторонні сервіси без необхідності вносити зміни в серверну логіку.

Модуль реєстрації нових користувачів (рис. 3.14) об'єднує стандартну процедуру створення облікового запису з процесом внесення біометричних даних у систему.

Рис. 3.14 – Форма реєстрації нового користувача

На відміну від форми входу, інтерфейс реєстрації надає користувачеві розширені можливості для надання еталонного зображення: окрім захоплення знімка з вебкамери в реальному часі, реалізовано опцію завантаження готового

файлу з локального сховища пристрою. Це архітектурне рішення дозволяє зменшити відсів користувачів у випадках, коли якість веб-камери є недостатньою для створення якісного первинного зліпка.

Успішна автентифікація відкриває доступ до особистого кабінету (Рис. 3.15). Базовий функціонал сторінки дозволяє звичайному користувачеві переглянути свій поточний статус та, за необхідності, скористатися правом на видалення власних персональних даних («право на забуття») згідно з вимогами GDPR.

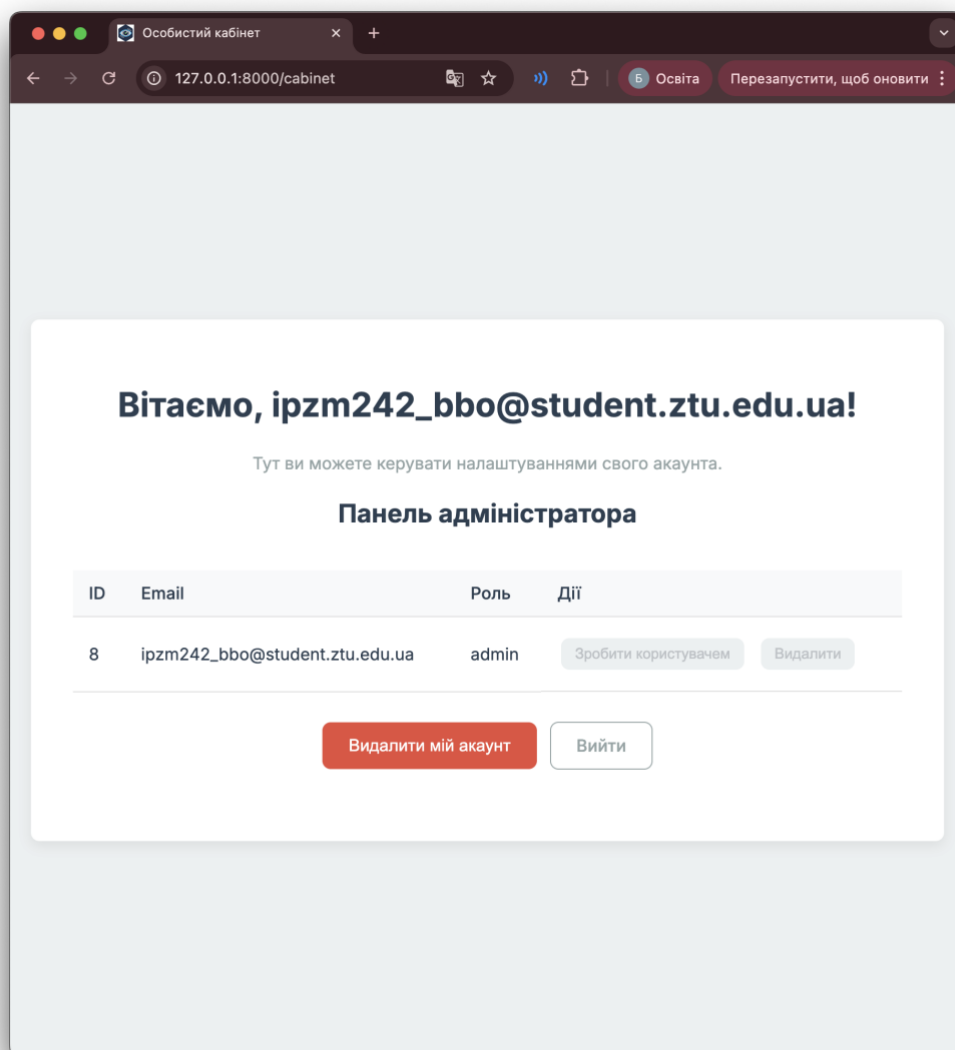


Рис. 3.15 – Панель керування та інтерфейс адміністратора системи

Окремо в системі реалізовано рольову модель доступу. Алгоритм реєстрації запрограмовано таким чином, що перший зареєстрований користувач

автоматично отримує роль адміністратора. Це забезпечує наявність суперкористувача одразу після розгортання системи.

Таким чином, розроблений клієнтський додаток не лише реалізує сценарії біометричної верифікації, а й забезпечує необхідні інструменти для адміністрування та підтримки життєвого циклу користувачів у системі.

Висновки до розділу 3

У третьому розділі здійснено програмну реалізацію системи біометричної верифікації, в основу якої покладено високоефективний стек технологій, що забезпечує асинхронну обробку запитів та необхідну продуктивність для роботи в режимі реального часу.

Ключовим архітектурним рішенням стало перенесення обчислювально складних операцій пошуку «найближчого сусіда» з рівня додатку на рівень бази даних. Завдяки використанню розширення pgvector та імплементації індексу HNSW [27], вдалося досягти логарифмічної складності пошуку, що гарантує миттєву реакцію системи навіть при роботі з надвеликими масивами біометричних векторів.

Окрему увагу приділено питанням безпеки та розмежування прав доступу. У системі реалізовано захист облікових даних користувачів із використанням адаптивного алгоритму хешування bcrypt, який унеможливорює атаки методом повного перебору. Також впроваджено рольову модель доступу, що передбачає автоматичне призначення прав адміністратора та наявність спеціалізованого інтерфейсу для керування користувачами.

За результатами експериментального порівняння підтверджено перевагу моделі ArcFace, яка продемонструвала точність розпізнавання на рівні 92% при середньому часі інференсу 47 мс. Аналіз алгоритмічної упередженості на базі датасету FairFace засвідчив стабільність роботи системи для різних демографічних груп.

Практичним результатом розділу є розробка повнофункціонального вебдодатку, який не лише виконує функції біометричної ідентифікації, а й

забезпечує дотримання вимог конфіденційності, надаючи користувачам інструменти для повного видалення своїх персональних даних.

ВИСНОВКИ

У магістерській кваліфікаційній роботі успішно вирішено науково-прикладне завдання підвищення рівня захищеності інформаційних систем шляхом розробки та програмної реалізації системи біометричної верифікації користувачів на основі глибоких нейронних мереж. Проведений у ході дослідження аналіз предметної області засвідчив, що в умовах зростання кіберзагроз традиційні методи парольної аутентифікації вичерпали свій потенціал. Це обґрунтувало доцільність переходу до впровадження біометричних систем розпізнавання обличчя, які, порівняно з апаратними токенами чи SMS-інформуванням, забезпечують значно вищий рівень безпеки та покращений досвід користування.

На основі результатів порівняльного аналізу архітектур нейронних мереж, зокрема VGG-Face, FaceNet та ArcFace, було визначено, що оптимальним рішенням для використання у веб-середовищі є модель ArcFace. Доведено, що завдяки використанню функції втрат з адитивним кутовим відступом, ця архітектура дозволяє досягти кращої компактності внутрішньокласових кластерів на гіперсфері. Це є важливим фактором для забезпечення точної верифікації в складних умовах експлуатації, наприклад, при змінному освітленні.

В рамках роботи спроектовано архітектуру та виконано програмну реалізацію системи у вигляді високопродуктивного вебсервісу на базі мови програмування Python 3.11 та фреймворку FastAPI. Ключовою особливістю розробленого рішення стало перенесення обчислювального навантаження пошуку ідентичності з рівня додатку безпосередньо на рівень бази даних PostgreSQL. Завдяки інтеграції розширення pgvector та застосуванню індексу HNSW, вдалося досягти логарифмічної складності пошуку, що гарантує миттєвий відгук системи навіть при суттєвому масштабуванні кількості користувачів.

Окрему увагу в роботі приділено питанням безпеки та етики. У системі реалізовано комплексну модель захисту, що включає використання адаптивного

алгоритму хешування bcrypt для захисту облікових даних від атак повного перебору, а також рольову модель доступу з автоматичним наданням прав адміністратора та відповідною панеллю керування. Експериментальні дослідження підтвердили високу ефективність запропонованих рішень: точність верифікації на валідаційному наборі даних склала 92% при середньому часі обробки запиту 47 мс. Додаткове тестування на алгоритмічну упередженість з використанням збалансованого датасету FairFace засвідчило відсутність критичних відхилень у точності розпізнавання для різних демографічних груп. Практична цінність роботи полягає у створенні повністю готового до впровадження програмного продукту, який має інтуїтивно зрозумілий графічний інтерфейс та механізми захисту персональних даних, що відповідають вимогам регламенту GDPR, зокрема реалізують право користувача на повне видалення біометричного шаблону.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jain A. K., Ross A. A., Nandakumar K. Introduction to Biometrics. Springer, 2011. 336 p. DOI: 10.1007/978-0-387-77326-1
2. Sze V., Chen Y., Yang T., Emer J. S. Efficient Processing of Deep Neural Networks. Morgan & Claypool Publishers, 2020. 341 p. DOI: 10.2200/S01004ED1V01Y202004CAC050
3. Drozdowski P. et al. Demographic Bias in Biometrics: A Survey on an Emerging Challenge. IEEE Transactions on Technology and Society. 2020. Vol. 1, No. 2. P. 91–103. DOI: 10.1109/TTS.2020.2992344
4. Савіцький Р. С., Бабицький Б. О. Аналіз архітектур нейронних мереж для розпізнавання облич // Матеріали XVIII Міжнародної науково-практичної конференції «Інформаційні технології і автоматизація – 2025». Одеса: ОНТУ, 2025. С. 1032.
5. Grassi P. A. et al. Digital Identity Guidelines. NIST Special Publication 800-63-3. National Institute of Standards and Technology, 2017. DOI: 10.6028/NIST.SP.800-63-3
6. Kortli Y., Jridi M., Al Falou A., Atri M. Face Recognition Systems: A Survey. Sensors. 2020. Vol. 20, No. 2. P. 342. DOI: 10.3390/s20020342
7. Adjabi I., Ouahabi A., Benzaoui A., Taleb-Ahmed A. Past, Present, and Future of Face Recognition: A Review. Electronics. 2020. Vol. 9, No. 8. P. 1188. DOI: 10.3390/electronics9081188
8. Masi I., Wu Y., Hassner T., Natarajan P. Deep Face Recognition: A Survey. 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI). 2018. P. 471–478. DOI: 10.1109/SIBGRAPI.2018.00067
9. Viola P., Jones M. Rapid Object Detection using a Boosted Cascade of Simple Features. CVPR. 2001. Vol. 1. P. 511–518. DOI: 10.1109/CVPR.2001.990517
10. Parkhi O. M., Vedaldi A., Zisserman A. Deep Face Recognition. British Machine Vision Conference (BMVC). 2015. Vol. 1, No. 3. P. 41–52. URL: <http://www.bmva.org/bmvc/2015/papers/paper041/index.html>

11. Cao Q., Shen L., Xie W., Parkhi O. M., Zisserman A. VGGFace2: A dataset for recognising faces across pose and age. International Conference on Automatic Face & Gesture Recognition (FG 2018). 2018. P. 67–74. DOI: 10.1109/FG.2018.00020
12. Deng J., Guo J., Xue N., Zafeiriou S. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019. P. 4690–4699. DOI: 10.1109/CVPR.2019.00482
13. Hu G., Yang Y., Yi D., Kittler J. When Face Recognition Meets with Deep Learning: An Evaluation of Convolutional Neural Networks for Face Recognition. IEEE International Conference on Computer Vision Workshops. 2015. P. 142–150. DOI: 10.1109/ICCVW.2015.28
14. Zhang K., Zhang Z., Li Z., Qiao Y. Joint Face Detection and Alignment Using Multi-task Cascaded Convolutional Networks. IEEE Signal Processing Letters. 2016. Vol. 23, No. 10. P. 1499–1503. DOI: 10.1109/LSP.2016.2603342
15. Kärkkäinen K., Joo J. FairFace: Face Attribute Dataset for Balanced Race, Gender, and Age. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). 2021. P. 1548–1558. URL: https://openaccess.thecvf.com/content/WACV2021/html/Karkkainen_FairFace_Face_Attribute_Dataset_for_Balanced_Race_Gender_and_Age_WACV_2021_paper.html
16. Pedregosa F. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011. Vol. 12. P. 2825–2830. URL: <https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html>
17. Bianco S., Mazzini D., Schettini R. Deep Visualization for Face Recognition. European Conference on Computer Vision (ECCV). 2016. P. 34–45. DOI: 10.1007/978-3-319-48881-3_3
18. Sagonas C. et al. 300 Faces In-the-wild Challenge: Database and results. Image and Vision Computing. 2016. Vol. 47. P. 3–18. DOI: 10.1016/j.imavis.2015.12.002

19. Python 3.11 Release Notes [Электронный ресурс]. Python Software Foundation, 2022. URL: <https://docs.python.org/3/whatsnew/3.11.html>
20. Bednarz B., Miłosz M. Benchmarking the performance of Python web frameworks. Journal of Computer Sciences Institute. 2025. Vol. 36. P. 336–341. DOI: 10.35784/jcsi.7738
21. Pydantic Documentation [Электронный ресурс]. URL: <https://docs.pydantic.dev/>
22. Bradski G. The OpenCV Library. Dr. Dobb's Journal of Software Tools. 2000. Vol. 25. P. 120–123. URL: <https://opencv.org/>
23. Serengil S. I., Ozpinar A. LightFace: A Hybrid Deep Face Recognition Framework. 2020 Innovations in Intelligent Systems and Applications Conference (ASYU). Istanbul, 2020. P. 1–5. DOI: 10.1109/ASYU50717.2020.9259802
24. Huang G. B., Mattar M., Berg T., Learned-Miller E. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition. Marseille, France, 2008. URL: <http://www.cs.umass.edu/lfw/lfw.pdf>
25. Hunter J. D. Matplotlib: A 2D graphics environment. Computing in Science & Engineering. 2007. Vol. 9, No. 3. P. 90–95. DOI: 10.1109/MCSE.2007.55
26. PostgreSQL Global Development Group. PostgreSQL 16 Documentation: pgvector [Электронный ресурс]. URL: <https://github.com/pgvector/pgvector>
27. Malkov Y. A., Yashunin D. A. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2020. Vol. 42, No. 4. P. 824–836. DOI: 10.1109/TPAMI.2018.2889473
28. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. Internet Engineering Task Force (IETF), 2015. DOI: 10.17487/RFC7519
29. Regulation (EU) 2016/679 of the European Parliament and of the Council (GDPR). Official Journal of the European Union. 2016. L 119. P. 1–88.

30. Про захист персональних даних: Закон України від 01.06.2010 № 2297-VI.
Відомості Верховної Ради України. 2010. № 34. Ст. 481.

ДОДАТКИ

Фрагмент коду алгоритму порівняльного аналізу швидкодії та точності нейронних мереж

```

from deepface import DeepFace
from sklearn.metrics import accuracy_score, roc_auc_score, roc_curve
from sklearn.metrics.pairwise import cosine_similarity
import tensorflow_datasets as tfds
import random
import sys
import numpy as np
from collections import defaultdict
from memory_profiler import memory_usage
import matplotlib.pyplot as plt
import time
import tracemalloc

def calcTime(func):
    def wrapper(*args, **kwargs):
        start = time.perf_counter()
        result = func(*args, **kwargs)
        end = time.perf_counter()
        print(f"🕒 {func.__name__} виконалась за {end - start:.4f} сек")
        return result
    return wrapper

print("sys.prefix:", sys.prefix)
print("base_prefix:", sys.base_prefix)
print("Is venv active?", sys.prefix != sys.base_prefix)

if sys.prefix == sys.base_prefix:
    exit("Should be run with venv")

pairs = []

lfw = tfds.load("lfw", split="train", as_supervised=True, shuffle_files=True)
lfw_list = list(tfds.as_numpy(lfw))

label_to_imgs = defaultdict(list)

for label, img in lfw_list:
    label_to_imgs[label].append(img)

def get_positive_pair():
    while True:
        label, imgs = random.choice(list(label_to_imgs.items()))
        if len(imgs) >= 2:
            return random.sample(imgs, 2), label

def get_negative_pair():
    label1, imgs1 = random.choice(list(label_to_imgs.items()))
    label2, imgs2 = random.choice(list(label_to_imgs.items()))
    while label1 == label2:
        label2, imgs2 = random.choice(list(label_to_imgs.items()))
    return random.choice(imgs1), random.choice(imgs2), (label1, label2)

@calcTime
def generate_pairs(n=10):

```

```

positive_pairs = []
negative_pairs = []
for _ in range(n):
    pos_pair, label = get_positive_pair()
    positive_pairs.append((pos_pair[0], pos_pair[1], True))
    neg_img1, neg_img2, labels = get_negative_pair()
    negative_pairs.append((neg_img1, neg_img2, False))
return positive_pairs + negative_pairs

def test_model(model, pairs):
    results = {}

    y_true = []
    y_pred_labels = []
    y_scores = []

    total_time = 0
    peak_mem = 0

    for img1, img2, is_same in pairs:
        start_time = time.time()
        tracemalloc.start()

        try:
            result = DeepFace.verify(img1_path=img1, img2_path=img2,
model_name=model, enforce_detection=False)
            pred_same = result["verified"]
            distance = result["distance"]

            score = -distance

            y_true.append(1 if is_same else 0)
            y_pred_labels.append(1 if pred_same else 0)
            y_scores.append(score)

        except Exception as e:
            print(f"Помилка для {model}: {e}")

        end_time = time.time()
        current, peak = tracemalloc.get_traced_memory()
        tracemalloc.stop()

        total_time += end_time - start_time
        peak_mem = max(peak_mem, peak)

    acc = accuracy_score(y_true, y_pred_labels)
    try:
        auc = roc_auc_score(y_true, y_scores)
    except ValueError:
        auc = None
    results[model] = {"accuracy": acc, "auc": auc}

    fpr, tpr, _ = roc_curve(y_true, y_scores)
    plt.plot(fpr, tpr, label=f"{model} (AUC={auc:.2f})")

    print(f"\nМодель: {model}")
    print(f"Accuracy: {acc:.2f}")
    print(f"AUC: {auc:.2f}")
    print(f"Середній час на перевірку: {total_time / len(pairs):.4f} сек")
    print(f"Пікова пам'ять: {peak_mem / 1024 / 1024:.2f} MB")

    return {"accuracy": acc, "auc": auc, "avg_time": total_time / len(pairs),
"peak_mem": peak_mem / 1024 / 1024}

```

```

pairs = generate_pairs(n=50)
models = ["VGG-Face", "Facenet", "ArcFace"]

results = {}
for model in models:
    results[model] = test_model(model, pairs)

plt.plot([0, 1], [0, 1], "k--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC-криві моделей")
plt.legend()
plt.show()

```

Реалізація методики оцінки демографічної упередженості моделі ArcFace

```

import os
import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from deepface import DeepFace
from itertools import combinations
import random

DATASET_PATH = "../datasets/fairface-img-margin025-trainval"
LABEL_FILE_PATH = "fairface_label_val.csv"
MODEL_NAME = "ArcFace"
DETECTOR = "mtcnn"
DISTANCE_METRIC = "cosine"

class BiasAnalyzer:
    def __init__(self, img_folder_path, label_file_path):
        self.img_folder_path = os.path.abspath(img_folder_path)
        self.label_file_path = label_file_path
        self.metadata = []
        self.embeddings = {}

    def load_data(self, limit=None):
        print(f"--- Завантаження даних з {self.label_file_path} ---")

        if not os.path.exists(self.label_file_path):
            print(f"Помилка: Файл з мітками '{self.label_file_path}' не  
знайдено.", file=sys.stderr)
            return False

        df = pd.read_csv(self.label_file_path)

        df['path'] = df['file'].apply(lambda x:
os.path.join(self.img_folder_path, x))

        df = df[df['path'].apply(os.path.exists)]

        if len(df) == 0:
            print("Помилка: Не знайдено жодного зображення. Перевірте, що:",
file=sys.stderr)
            print(f"  1. Шлях '{self.img_folder_path}' є правильним.",
file=sys.stderr)
            print(f"  2. Всередині нього є папка 'val' з зображеннями.",
file=sys.stderr)

```

```

        return False

    if limit:
        df = df.sample(n=min(limit, len(df)), random_state=42)

    self.metadata = df.to_dict('records')
    print(f"Завантажено {len(self.metadata)} зображень.")
    return True

def get_embeddings(self):
    print(f"--- Генерація векторів (Model: {MODEL_NAME}) ---")
    valid_data = []

    for idx, item in enumerate(self.metadata):
        try:
            embedding_obj = DeepFace.represent(
                img_path=item['path'],
                model_name=MODEL_NAME,
                detector_backend=DETECTOR,
                enforce_detection=False,
                align=True
            )
            vector = embedding_obj[0]["embedding"]
            self.embeddings[item['path']] = vector
            valid_data.append(item)

        except Exception as e:
            print(f"Помилка обробки {item['path']}: {e}")

        if (idx + 1) % 10 == 0:
            print(f"Оброблено {idx + 1} / {len(self.metadata)}")

    self.metadata = valid_data
    print("Векторизація завершена.")

def evaluate_bias(self, group_by='race'):
    print(f"--- Аналіз упереджень за групою '{group_by}' ---")
    df = pd.DataFrame(self.metadata)
    results = []

    if group_by not in df.columns:
        print(f"Помилка: колонка '{group_by}' відсутня в даних.",
file=sys.stderr)
        return pd.DataFrame()

    groups = df[group_by].unique()

    for group in groups:
        group_df = df[df[group_by] == group]
        if len(group_df) < 2: continue

        paths = group_df['path'].tolist()

        pairs = list(combinations(paths, 2))
        if len(pairs) > 500:
            pairs = random.sample(pairs, 500)

        distances = []
        for img1, img2 in pairs:
            v1 = self.embeddings.get(img1)
            v2 = self.embeddings.get(img2)
            if v1 is not None and v2 is not None:
                cos_sim = np.dot(v1, v2) / (np.linalg.norm(v1) *
np.linalg.norm(v2))

```

```

        distance = 1 - cos_sim
        distances.append(distance)

    if distances:
        avg_dist = np.mean(distances)
        results.append({"Group": group, "Avg_Distance": avg_dist,
"Samples": len(distances)})

    return pd.DataFrame(results)

def plot_results(df_results, group_by):
    if df_results.empty:
        print("Немає даних для побудови графіка.")
        return

    plt.figure(figsize=(10, 6))
    df_results = df_results.sort_values('Avg_Distance', ascending=False)

    colors = plt.cm.viridis(np.linspace(0, 1, len(df_results)))
    bars = plt.bar(df_results['Group'], df_results['Avg_Distance'],
color=colors)

    plt.title(f'Середня внутрішньогрупова дистанція (Модель: {MODEL_NAME})',
fontsize=14)
    plt.xlabel(f'Група ({group_by.capitalize()})', fontsize=12)
    plt.ylabel('Середня Cosine Distance (менше = краще)', fontsize=12)
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.xticks(rotation=45, ha="right")

    for bar in bars:
        height = bar.get_height()
        plt.text(bar.get_x() + bar.get_width() / 2., height, f'{height:.4f}',
ha='center', va='bottom')

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    analyzer = BiasAnalyzer(img_folder_path=DATASET_PATH,
label_file_path=LABEL_FILE_PATH)

    if analyzer.load_data(limit=500):
        analyzer.get_embeddings()

        bias_stats_race = analyzer.evaluate_bias(group_by='race')
        print("\nРезультати аналізу за расою:")
        print(bias_stats_race)

        bias_stats_gender = analyzer.evaluate_bias(group_by='gender')
        print("\nРезультати аналізу за статтю:")
        print(bias_stats_gender)

        if bias_stats_race is not None and not bias_stats_race.empty:
            plot_results(bias_stats_race, group_by='race')

        if bias_stats_gender is not None and not bias_stats_gender.empty:
            plot_results(bias_stats_gender, group_by='gender')

```

Програмна реалізація серверної частини системи розпізнавання

```
import os
import numpy as np
import cv2
from fastapi import FastAPI, Request, UploadFile, File, Form, HTTPException,
Depends, status
from fastapi.security import OAuth2PasswordBearer
from fastapi.staticfiles import StaticFiles
from fastapi.templating import Jinja2Templates
from fastapi.responses import HTMLResponse, JSONResponse
from dotenv import load_dotenv
import asyncpg
from deepface import DeepFace
import json
from pydantic import BaseModel, EmailStr
from passlib.context import CryptContext
import secrets
from datetime import datetime, timedelta
from jose import JWTError, jwt
from enum import Enum

load_dotenv()

SECRET_KEY = os.getenv("SECRET_KEY", secrets.token_urlsafe(32))
ALGORITHM = "HS256"
ACCESS_TOKEN_EXPIRE_MINUTES = 30

pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")

class UserRole(str, Enum):
    user = "user"
    admin = "admin"

class UpdateRoleRequest(BaseModel):
    role: UserRole

app = FastAPI(title="Async FastAPI Server Example")
templates = Jinja2Templates(directory="templates")
app.mount("/static", StaticFiles(directory="static"), name="static")

oauth2_scheme = OAuth2PasswordBearer(tokenUrl="token")

DB_POOL = None
MODEL_NAME = "ArcFace"
ARCFACE_THRESHOLD = 0.68

class Token(BaseModel):
    access_token: str
    token_type: str

class TokenData(BaseModel):
    user_id: int | None = None
    role: UserRole | None = None

def verify_password(plain_password, hashed_password):
    return pwd_context.verify(plain_password, hashed_password)

def get_password_hash(password):
    return pwd_context.hash(password)

def create_access_token(data: dict, expires_delta: timedelta | None = None):
    to_encode = data.copy()
```

```

if expires_delta:
    expire = datetime.utcnow() + expires_delta
else:
    expire = datetime.utcnow() + timedelta(minutes=15)
to_encode.update({"exp": expire})
encoded_jwt = jwt.encode(to_encode, SECRET_KEY, algorithm=ALGORITHM)
return encoded_jwt

async def setup_db_connection(conn):
    await conn.set_type_codec(
        'vector',
        encoder=lambda v: str(list(v)),
        decoder=lambda s: np.array(json.loads(s)),
        schema='public'
    )

@app.on_event("startup")
async def startup_handler():
    global DB_POOL
    database_url = os.getenv("DATABASE_URL")
    if not database_url:
        raise Exception("DATABASE_URL environment variable not set")
    DB_POOL = await asyncpg.create_pool(dsn=database_url,
    init=setup_db_connection)

    print("Loading face recognition model...")
    DeepFace.build_model(MODEL_NAME)
    print("Model loaded successfully.")

@app.on_event("shutdown")
async def shutdown_db_client():
    if DB_POOL:
        await DB_POOL.close()

async def get_current_user(token: str = Depends(oauth2_scheme)):
    credentials_exception = HTTPException(
        status_code=status.HTTP_401_UNAUTHORIZED,
        detail="Could not validate credentials",
        headers={"WWW-Authenticate": "Bearer"},
    )
    try:
        payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
        user_id: int = int(payload.get("sub"))
        role: str = payload.get("role")
        if user_id is None or role is None:
            raise credentials_exception
        token_data = TokenData(user_id=user_id, role=UserRole(role))
    except (JWTError, ValueError, TypeError):
        raise credentials_exception

    async with DB_POOL.acquire() as connection:
        user = await connection.fetchrow("SELECT id, email, role FROM users
WHERE id = $1", token_data.user_id)
        if user is None:
            raise credentials_exception
    return dict(user)

def get_current_admin_user(current_user: dict = Depends(get_current_user)):
    if current_user['role'] != UserRole.admin:
        raise HTTPException(status_code=403, detail="The user doesn't have
enough privileges")
    return current_user

@app.get("/health", status_code=200)

```

```

def health_check():
    return {"status": "ok"}

@app.get("/register", response_class=HTMLResponse)
async def register_page(request: Request):
    return templates.TemplateResponse("register.html", {"request": request})

@app.post("/register")
async def register(
    email: str = Form(...),
    password: str = Form(...),
    face_image: UploadFile = File(...)
):
    try:
        image_data = await face_image.read()
        np_arr = np.frombuffer(image_data, np.uint8)
        img = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
        if img is None:
            raise HTTPException(status_code=400, detail="Не вдалося обробити
зображення.")
        except Exception as e:
            raise HTTPException(status_code=400, detail=f"Помилка читання або
обробки зображення: {e}")

        try:
            embedding_obj = DeepFace.represent(img_path=img, model_name=MODEL_NAME,
enforce_detection=True)
            embedding = embedding_obj[0]['embedding']
        except ValueError as e:
            raise HTTPException(status_code=400, detail=f"Не вдалося знайти обличчя
на зображенні: {e}")
        except Exception as e:
            raise HTTPException(status_code=500, detail=f"Помилка генерації
векторного представлення обличчя: {e}")

        password_hash = get_password_hash(password)

        async with DB_POOL.acquire() as connection:
            existing_user_by_email = await connection.fetchrow("SELECT email FROM
users WHERE email = $1", email)
            if existing_user_by_email:
                raise HTTPException(status_code=409, detail="Користувач з такою
електронною поштою вже існує.")

            closest_match = await connection.fetchrow("SELECT face_embedding <=> $1
AS distance FROM users ORDER BY distance LIMIT 1;", embedding)
            if closest_match and closest_match['distance'] <= ARCFACE_THRESHOLD:
                raise HTTPException(status_code=409, detail="Користувач з таким
обличчям вже існує.")

            user_count = await connection.fetchval("SELECT COUNT(*) FROM users")
            role = UserRole.admin if user_count == 0 else UserRole.user

            await connection.execute(
                "INSERT INTO users (email, password_hash, face_embedding, role)
VALUES ($1, $2, $3, $4)",
                email,
                password_hash,
                embedding,
                role
            )

    return JSONResponse(content={"status": "success", "message": f"Користувач
{email} успішно зареєстрований з роллю {role}."}, status_code=201)

```

```

@app.get("/login", response_class=HTMLResponse)
async def login_page(request: Request):
    return templates.TemplateResponse("login.html", {"request": request})

@app.post("/token", response_model=Token)
async def login_for_access_token(
    type: str = Form(...),
    email: str = Form(None),
    password: str = Form(None),
    image: UploadFile = File(None)
):
    user = None
    if type == "biometric":
        if not image:
            raise HTTPException(status_code=400, detail="Для біометричного входу
потрібне зображення.")
        try:
            image_data = await image.read()
            np_arr = np.frombuffer(image_data, np.uint8)
            img = cv2.imdecode(np_arr, cv2.IMREAD_COLOR)
            if img is None:
                raise HTTPException(status_code=400, detail="Не вдалося обробити
зображення з файлу.")
            except Exception as e:
                raise HTTPException(status_code=400, detail=f"Не вдалося прочитати
файл зображення: {e}")

            try:
                embedding_obj = DeepFace.represent(img_path=img,
model_name=MODEL_NAME, enforce_detection=True)
                login_embedding = embedding_obj[0]['embedding']
            except ValueError:
                raise HTTPException(status_code=400, detail="Не вдалося знайти
обличчя на зображенні для входу.")
            except Exception as e:
                raise HTTPException(status_code=500, detail=f"Помилка обробки
зображення: {e}")

            async with DB_POOL.acquire() as connection:
                query = "SELECT id, email, role, face_embedding <=> $1 AS distance
FROM users ORDER BY distance LIMIT 1;"
                closest_match = await connection.fetchrow(query, login_embedding)

                if closest_match and closest_match['distance'] <= ARCFACE_THRESHOLD:
                    user = closest_match

        elif type == "password":
            if not email or not password:
                raise HTTPException(status_code=400, detail="Для входу за паролем
потрібні email та пароль.")

            async with DB_POOL.acquire() as connection:
                db_user = await connection.fetchrow("SELECT id, email, role,
password_hash FROM users WHERE email = $1", email)
                if db_user and verify_password(password, db_user['password_hash']):
                    user = db_user

    if not user:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Неправильний email, пароль або обличчя.",
            headers={"WWW-Authenticate": "Bearer"},
        )

```

```

        access_token_expires = timedelta(minutes=ACCESS_TOKEN_EXPIRE_MINUTES)
        access_token = create_access_token(
            data={"sub": str(user['id']), "role": user['role']},
            expires_delta=access_token_expires
        )
        return {"access_token": access_token, "token_type": "bearer"}

@app.get("/cabinet", response_class=HTMLResponse)
async def cabinet(request: Request):
    return templates.TemplateResponse("cabinet.html", {"request": request})

@app.get("/users/me", response_model=dict)
async def read_users_me(current_user: dict = Depends(get_current_user)):
    return current_user

@app.post("/delete_account")
async def delete_account(current_user: dict = Depends(get_current_user)):
    user_id_to_delete = current_user['id']
    async with DB_POOL.acquire() as connection:
        result = await connection.execute("DELETE FROM users WHERE id = $1",
user_id_to_delete)
        if result == "DELETE 1":
            return JSONResponse(content={"status": "success", "message": "Акаунт
успішно видалено."})
        else:
            raise HTTPException(status_code=404, detail="Користувача не
знайдено.")

@app.get("/users", response_model=list[dict])
async def get_users(current_user: dict = Depends(get_current_admin_user)):
    async with DB_POOL.acquire() as connection:
        rows = await connection.fetch("SELECT id, email, role FROM users")
        return [{"id": row['id'], "email": row['email'], "role": row['role']}
for row in rows]

@app.put("/users/{user_id}/role", status_code=200)
async def update_user_role(user_id: int, data: UpdateRoleRequest, current_admin:
dict = Depends(get_current_admin_user)):
    if user_id == current_admin['id']:
        raise HTTPException(status_code=400, detail="Адміністратор не може
змінити власну роль.")

    async with DB_POOL.acquire() as connection:
        result = await connection.execute("UPDATE users SET role = $1 WHERE id =
$2", data.role, user_id)
        if result == "UPDATE 1":
            return {"status": "success", "message": f"Роль користувача оновлено
на {data.role}."}
        else:
            raise HTTPException(status_code=404, detail="Користувача не
знайдено.")

@app.delete("/users/{user_id}", status_code=200)
async def delete_user_by_id(user_id: int, current_admin: dict =
Depends(get_current_admin_user)):
    if user_id == current_admin['id']:
        raise HTTPException(status_code=400, detail="Адміністратор не може
видалити самого себе.")

    async with DB_POOL.acquire() as connection:
        result = await connection.execute("DELETE FROM users WHERE id = $1",
user_id)
        if result == "DELETE 1":

```

```
        return {"status": "success", "message": "Користувача успішно  
видалено."}  
    else:  
        raise HTTPException(status_code=404, detail="Користувача не  
знайдено.")  
  
@app.get("/")  
async def root(request: Request):  
    return templates.TemplateResponse("index.html", {"request": request})
```