Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Навчально-науковий інститут комп’ютерних систем
Кафедра інженерії програмного забезпечення

Безрученко Ярослав Віталійович,
студент групи АС-172

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Програмна система машинного зору для біометричної ідентифікації осіб за двовимірними зображеннями з використанням методів штучного інтелекту.
Підтема 1. Серверна частина

Спеціальність:
121 – Інженерія програмного забезпечення

Освітня програма:
Інженерія програмного забезпечення

Керівник:
Крісілов Віктор Анатолійович,
доктор технічних наук, професор

Одеса – 2022

ЗМІСТ

Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Навчально-науковий інститут комп’ютерних систем
Кафедра інженерії програмного забезпечення

Рівень вищої освіти: другий (магістерський)
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Комлева Н. О.
«___» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Безрученко Ярослава Віталійовича, група АС-172

1. Тема роботи: Програмна система машинного зору для біометричної ідентифікації осіб за двовимірними зображеннями з використанням методів штучного інтелекту. Підтема 1. Серверна частина

Керівник роботи: Крісілов Віктор Анатолійович, доктор технічних наук, професор

затверджені наказом ректора від «20» жовтня 2022 р. № 399-в.

2. Зміст роботи: критичний аналіз існуючих рішень, специфікація вимог, розробка системи, проектування серверної частини програмної системи, програмна реалізація серверу, тестування системи

3. Перелік ілюстративного матеріалу: згідно зі слайдами презентації

4. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Дата видачі завдання: «01» вересня 2022р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	Критичний аналіз існуючих рішень	10.09.2022	виконав
2	Розробка серверної частини системи	20.09.2022	виконав
3	Специфікація вимог до серверної частини	10.10.2022	виконав
4	Проектування серверної частини системи	20.10.2022	виконав
5	Програмна реалізація серверної частини	02.11.2022	виконав
6	Тестування роботи системи	20.11.2022	виконав
7	Оформлення документації	05.12.2022	виконав

Здобувач вищої освіти_____ Я. В. Безрученко

Керівник роботи_____ В. А. Крісілов

АНОТАЦІЯ

Метою роботи є підвищення точності біометричної ідентифікації осіб за двовимірними зображеннями з використанням методів штучного інтелекту. В даній кваліфікаційній роботі розроблено серверну частину програмної системи машинного зору, а також обрано топологію та виконано реалізацію згорткової нейронної мережі.

Методи розробки засновані на використанні нейронних мереж, які виконують біометричну ідентифікацію за зображеннями обличчя людини та за її соматотипом. Розробка виконана на основі Android з використанням мови програмування Kotlin та бази даних PostgreSQL 12.

Ключові слова: машинний зір, архітектура клієнтської частини, біометрична ідентифікація, штучний інтелект, Android, Kotlin.

ABSTRACT

The aim of the work is to increase the accuracy of biometric identification of persons based on two-dimensional images using artificial intelligence methods. In this qualification work, the server part of the machine vision software system was developed, and the topology was chosen and the convolutional neural network was implemented.

Processing methods are based on the use of neural networks that perform biometric identification based on images of a person's face and their somatotype. The development is based on Android using the Kotlin programming language and the PostgreSQL 12 database.

Keywords: machine vision, client-side architecture, biometric identification, artificial intelligence, Android, Kotlin.

ВСТУП

«Біометрія – це ідентифікація людини за унікальними, властивими тільки їй біологічними ознаками. Сьогодні експлуатується вже більше десятка різних біометричних ознак. Причому для найпоширеніших з них (відбитки пальців і райдужна оболонка ока) існує безліч різних за принципом дії сканерів. Так що користувачам, що вирішили використати біометричну ідентифікацію, є із чого вибрати.» [1]

Питання, пов’язані з ідентифікацією осіб, наприклад «Це та особа, за яку він або вона себе видає?», «Чи був цей заявник тут раніше?», «Чи слід надати цій особі доступ до нашої системи?» щодня мільйони разів запитують організації, що працюють у сфері фінансових послуг, охорони здоров’я, електронної комерції, телекомунікацій та уряду. Насправді шахрайство з особистими даними під час виплат соціальної допомоги, операцій з кредитними картками, дзвінків по мобільному телефону та зняття коштів із банкоматів становить понад 6 мільярдів доларів щороку.

З цієї причини все більше організацій шукають автоматизовані системи автентифікації, щоб підвищити задоволеність клієнтів і ефективність роботи, а також заощадити важливі ресурси. Крім того, у міру того, як люди стають більш зв’язаними за допомогою електроніки, здатність створити високоточну автоматичну систему ідентифікації особи стає значно важливішою.

Персональна ідентифікація - це процес асоціювання конкретної особи з ідентичністю. Ідентифікація може здійснюватися у формі верифікації (також відомої як автентифікація), яка передбачає підтвердження автентичності заявленої особистості («Чи я є тим, за кого себе виявляю?»), або розпізнавання (також відомої як ідентифікація), що передбачає визначення особистості задана особа з бази даних осіб, відомих системі («Хто я?»). Підходи до автоматичної ідентифікації особистості, засновані на знаннях і на основі маркерів, були двома традиційними

методами, які широко використовувалися [8]. Підходи на основі токенів використовують щось, що ви маєте для ідентифікації особи, наприклад паспорт, водійські права, посвідчення особи, кредитну картку або ключі. Підходи, засновані на знаннях, використовують щось, що ви знаєте, для ідентифікації особи, наприклад пароль або персональний ідентифікаційний номер (PIN). Оскільки ці традиційні підходи не ґрунтуються на будь-яких притаманних атрибутах особи для ідентифікації особистості, вони страждають від очевидних недоліків: токени можуть бути втрачені, викрадені, забуті чи згублені, а PIN-код може бути забутий дійсним користувачем або вгадав самозванець. (Як не дивно, але приблизно 25% людей пишуть свій PIN-код на своїй картці банкомату, таким чином порушуючи захист, який пропонує PIN-код у разі викрадення карток банкомату!) Оскільки підходи, засновані на знаннях, і підходи, що базуються на маркерах, не можуть розрізнити між собою уповноважена особа та самозванець, який шахрайським шляхом отримує маркер або знання уповноваженої особи, вони є незадовільними засобами для досягнення вимог безпеки нашого електронно взаємопов’язаного інформаційного суспільства.

Біометрична ідентифікація означає ідентифікацію особи на основі її фізіологічних та/або поведінкових особливостей (біометричні ідентифікатори). Він асоціює/роз’єднує особу з попередньо визначеною ідентичністю/ідентичністями на основі того, як вона є або що вона робить. Оскільки багато фізіологічних або поведінкових характеристик є відмінними для кожної людини, біометричні ідентифікатори за своєю суттю надійніші та ефективніші, ніж методи, засновані на знаннях і маркерах, у розрізненні уповноваженої особи від шахрая.

Об’єктом дослідження є біометрична ідентифікація осіб за двовимірними зображеннями.

Предметом є серверна частина програмної системи машинного зору.

Метою роботи є підвищення точності біометричної ідентифікації осіб за двовимірними зображеннями з використанням методів штучного інтелекту.

В даній кваліфікаційній роботі розроблено серверну частину програмної системи машинного зору, а також обрано топологію та виконано реалізацію згорткової нейронної мережі.

Розглянемо задачі, що необхідно виконати для досягнення поставленої мети:

- проведення критичного аналізу існуючих рішень;
- вибір базових метрик біометричної ідентифікації;
- вибір топології згорткової нейронної мережі для проведення ідентифікації;
- специфікація вимог до серверної частини системи;
- розробка архітектури серверної частини;
- програмна реалізація функціоналу серверної частини системи;
- проведення тестування системи.

Робота містить 6 розділів.У першому розділі було розглянуто сучасний стан проблеми біометричної ідентифікації та проведено аналіз алгоритмів-аналогів. У другому розділі була проведена розробка для серверної частини системи, а саме були побудовані дерева проблем та рішень, обрано топологію нейронної мережі для проведення біометричної ідентифікації за зображеннями обличчя та геометрією тіла людини. У третьому розділі було описано функціональні та нефункціональні вимоги до розроблюваного програмного продукту. У четвертому розділі виконано проектування серверної частини системи, а саме основні використовувані об’єкти і програмне забезпечення. У п’ятому розділі було створено програмну реалізацію серверної частини системи, у шостому – проведено тестування та описано контрольний приклад.

1 КРИТИЧНИЙ АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Сучасний стан проблеми біометричної ідентифікації

В останні роки цифрова обробка зображень та їх цифровий аналіз набувають великої популярності та застосовуються при вирішенні різних прикладних задач. Можливість застосування «комп'ютерного зору» багатогранна, а вирішення завдань у сфері може бути перенесено зовсім не пов'язану з нею прикладну область.

Комп'ютерний зір (машинний зір) – сукупність програмно-технічних засобів, що забезпечують зчитування в цифровій формі відеозображень, їх обробку та видачу результату у формі, придатній для практичного застосування в реальному масштабі часу. Обробці та аналізу зображень присвячена чимала кількість робіт, проте «універсального» рішення для будь-якого завдання на даний момент не створено. Кожна прикладна сфера накладає на рішення свої особливі умови.

Біометрична система, по суті, є системою розпізнавання образів, яка ідентифікує особу шляхом встановлення автентичності певної фізіологічної чи поведінкової характеристики користувача [2]. Логічно біометричну систему можна розділити на модуль реєстрації та модуль ідентифікації. На етапі реєстрації біометрична характеристика особи спочатку сканується біометричним датчиком для отримання цифрового представлення характеристики. Щоб полегшити зіставлення та зменшити вимоги до зберігання, цифрове представлення додатково обробляється екстрактором ознак для створення компактного, але виразного представлення, яке називається «шаблоном». Залежно від програми шаблон може зберігатися в центральній базі даних біометричної системи або бути записаний на магнітну картку чи смарт-картку, видану особі.

Під час фази розпізнавання біометричний зчитувач фіксує характеристики особи, яку потрібно ідентифікувати, і перетворює їх у цифровий формат, який далі обробляється екстрактором ознак для створення такого ж представлення, що й

шаблон. Результуюче представлення передається на пристрій зіставлення ознак, який порівнює його з шаблоном(ами), щоб встановити особу особи.

Ідеальна біометрія має бути універсальною, де кожна людина має свою характеристику; унікальний, де немає двох осіб, які мають спільну характеристику; постійний, де характеристика не повинна ні змінюватися, ні змінюватися; і колекційні, де характеристика легко представлена на сенсорі та легко піддається кількісному виміру.

Ступінь впровадження біометричних систем ідентифікації також залежить від того, наскільки технологія відповідає вимогам постачальників.

Але підвищений потенціал прибутку створює стимул для прискорення досліджень і вдосконалення технології.

З минулого досвіду зрозуміло одне: навіть якщо мотив виявиться короткочасним, чим більше буде прийнята технологія, тим важче буде обмежити її пізніше. Уявлення про те, що системи біометричної ідентифікації є розумним компромісом між конфіденційністю та безпекою, зробить їх прийнятними, навіть якщо ці компроміси не так явно на нашу користь.

Біометрія означає автоматичну ідентифікацію людини на основі її фізіологічних або поведінкових характеристик. Він забезпечує краще рішення для підвищених вимог безпеки нашого інформаційного суспільства, ніж традиційні методи ідентифікації, такі як паролі та PIN-коди. Оскільки біометричні датчики стають менш дорогими та мініатюрними, а громадськість усвідомлює, що біометрія насправді є ефективною стратегією захисту конфіденційності та від шахрайства, ця технологія, ймовірно, використовуватиметься майже в кожній транзакції, яка потребує автентифікації особистої особистості.

Біометрична ідентифікація породжує низку етичних питань, здебільшого зосереджених на концепції конфіденційності. Тут слід розрізняти два питання:

- чи викликає біометрична ідентифікація ті самі проблеми щодо конфіденційності даних, що й інші форми ідентифікації особистості?

- чи є проблеми з конфіденційністю, характерні для біометричних ідентифікаторів?

На захист ідеї, що біометрична технологія не створює серйозних проблем конфіденційності, пропонуються чотири основні аргументи:

1. Аргумент «технічних обмежень»: у великій популяції технологія має обмежені можливості для ідентифікації конкретної особи.

2. Аргумент «локальних обмежень»: інформація залишається локальною та обмеженою, оскільки не існує стандартів сумісності.

3. Аргумент «співпраці»: технологією не можна легко зловживати, оскільки ідентифікація вимагає співпраці.

4. Аргумент «безпеки»: алгоритми шаблонів є безпечними, оскільки постачальники біометричних даних зацікавлені в збереженні їх конфіденційності.

Представляють біометричні системи таку загрозу чи ні, залежить від того, чи справді їх можна використати, щоб виділити вас без вашої участі. Перше, на що тут слід звернути увагу, це аргумент «технічні обмеження». Які реальні та потенційні можливості технології? Загальновідомо, що технологія відбитків пальців є найнадійнішою. Багато постачальників стверджують, що рівень помилкових прийомів (false acceptance rate - FAR) і помилкових відхилень (false rejection rate - FRR) становить 0,01% або менше, що означає, що менше ніж один із 10 000 людей зіставляється з чужими відбитками пальців (FAR) або не зіставляється з їхніми власними відбитками пальців (FRR). Це означає, що якщо хтось хоче виділити вас (тобто, ви перебуваєте в «списку спостереження», на промисловому жаргоні), ваші відбитки пальців ідентифікуватимуть вас у 99,99 випадках зі 100.

Власники біометричних технологій можуть домовитися про протоколи обміну даними, не чекаючи галузевих стандартів. Але стандарти вже почали з'являтися. Стандарт зберігання відбитків пальців на водійських правах вже широко використовується.

1.2 Згорткові нейронні мережі як клас моделей машинного навчання

Довільна нейронна мережа складається з кількох взаємопов'язаних різних шарів, таких як вхідний шар, щонайменше один прихований шар та вихідний шар (рис. 1.2). Їх найкраще використовувати при виявленні об'єктів для розпізнавання образів, країв об’єктів – вертикальні та/чи горизонтальні, форми, кольору та текстури.

Приховані шари є шарами згортки. У даному типі нейронної мережі згорткові шари діє як фільтр, який спочатку отримує вхідні дані, перетворює їх, використовуючи певний алгоритм або функцію, і відправляє його на наступний шар. Основні параметри нейронів є вхідний (синій колір) та вихідний шар (зелений колір).

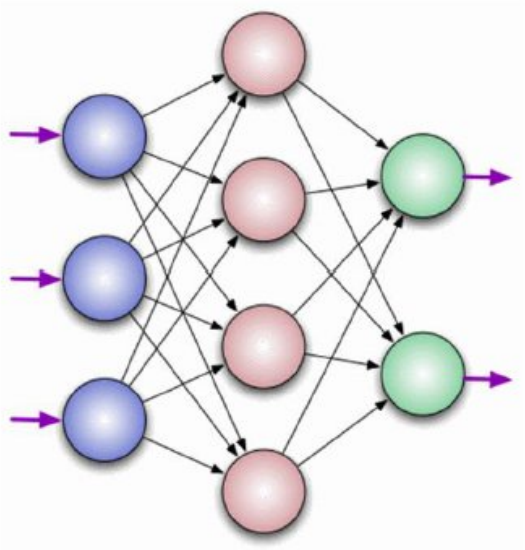

Рисунок 1.2 – Загальна модель нейронної мережі

«З великою кількістю згорткових шарів, щоразу, коли новий вхід відправляється на наступний згортковий шар, він змінюється по-різному. Наприклад, у згортковому шарі фільтр може ідентифікувати форму / колір у певній області, останній згортковий шар може класифікувати об'єкт.

У загальному випадку згорткова нейронна мережа складається з великої кількості шарів. На останніх етапах зазвичай використовується один або кілька повнозв'язкових шарів.

Згорткові нейронні мережі забезпечують часткову стійкість до змін масштабу, зсувів, поворотів, зміни ракурсу та інших спотворень. Згорткові нейронні мережі поєднують три архітектурні ідеї, для забезпечення інваріантності до зміни масштабу, повороту зсуву та просторових спотворень:

- локальні рецепторні поля (забезпечують локальну двовимірну зв'язність нейронів);
- загальні синоптичні коефіцієнти (забезпечують детектування деяких рис у будь-якому місці зображення та зменшують загальну кількість вагових коефіцієнтів);
- ієрархічна організація з просторовими підвиборками.» [3]

1.3 Базові метрики біометричної ідентифікації

На початку роботи було проаналізовано ключові точки на зображеннях обличчя, які обов’язково враховуються у будь-якому з алгоритмів біометричної ідентифікації. Тому в основу ключових лицьових точок полягла система з 10 базових метрик:

- висота рота: верхня верхня губа – низ нижньої губи;
- висота відкритого рота: низ верхньої губи – верх нижньої губи;
- куточок губ донизу: куточок рота – верх нижньої губи;
- куточок губ нагору: куточок рота – верх верхньої губи;
- ширина рота: лівий куточок рота – правий куточок рота;
- висота підборіддя: низ нижньої губи – підборіддя;
- ширина ока: верх ока – низ ока;
- висота брови: верхній центр брови – середина очей;

- внутрішній куточок брови: внутрішній кут брови – внутрішній кут ока;
- зовнішній куточок брови: зовнішній кут брови – зовнішній кут ока;

Ті метрики, які є несиметричними, розраховуються для лівої та правої половини окремо.

Сучасні дослідження демонструють, що доцільним також є застосування розпізнавання об’єкту (людини) за її соматотипом. Соматотип – це генетично обумовлена конституція людини, яка визначається на підставі антропометричних вимірів та не змінюється протягом життя людини. Це означає, що на соматотип не не впливають зміни тіла людини, обумовлені хворобами, старінням чи посиленими фізичними навантаженнями. Ряд досліджень показує, що зображення повного тіла людей розкривають їхній соматотип навіть після зміни одягу.

Значною перевагою цієї біометричної характеристики є те, що її можна легко зафіксувати, навіть на відстані, як повне зображення людини, зроблене стандартною 2D-камерою. Ця біометрична функція дозволяє розпізнавати людину також на відстані та під час руху. Для цього вивчаються та оцінюються два загальні сценарії: 1 - ідентифікації та 2 - перевірки.

У роботі використані загальнодоступні набори даних геометрії тіла людини, отриманих за допомогою чотирьох датчиків Microsoft Kinect NUI [4]. Ці датчики дозволили зробити виміри 17 показників геометрії тіла.

Для можливості використання даних геометрії тіла проводились різні сеанси, у яких людині пропонували нахилятись, робити повороти тощо, тобто вивчались усі можливі випадки, в тому числі в неідеальних умовах для подальшого розпізнавання.

Багаторазові виміри підтвердили високу надійність результатів, похибка вимірів склала менш ніж 1%. Це означає, що соматотип справді може бути цінною біометричною ознакою для розпізнавання особистості на відстані та в русі завдяки легкості отримання необхідних зображень.

У даній роботі для біометричної ідентифікації будуть використовуватись як зображеннях обличчя людини, так і дані геометрії тіла.

1.4 Аналіз аналогів

У 2014 році група дослідників з Каліфорнійського університету в Берклі розробила глибоку згортову мережу під назвою R-CNN (скорочено від regional-based convolutional neural network), яка може виявляти 80 різних типів об’єктів на зображеннях. Основний внесок мережі R-CNN полягає лише в тому, що вона виділяє функції на основі згорткової нейронної мережі (CNN). Крім цього, все інше схоже на загальний конвеєр виявлення об’єктів [5].

Згортова мережа RetinalNet часто використовується у задачах виявлення різних типів захворювань за двовимірним зображенням. Цей алгоритм машинного навчання виявився дуже точним у проблемах комп’ютерного зору [6].

Регіональні повністю згорткові мережі, або R-FCN, є типом регіонального детектора об’єктів. На відміну від попередніх регіональних детекторів об’єктів, таких як Fast/Faster R-CNN, які застосовують досить дорогу підмережу для кожного регіону сотні разів, R-FCN є повністю згортковим, майже всі обчислення розподіляються на все зображення.

Щоб досягти цього, R-FCN використовує чутливі до позиції оціночні карти для вирішення дилеми між трансляційною інваріантністю в класифікації зображень і трансляційною дисперсією у виявленні об’єктів. [7]

Попередньо розглянуті системи виявлення об’єктів перепрофілюють класифікатори або локалізатори для виконання цього виявлення. Вони застосовують модель до зображення в різних місцях і масштабах. Ділянки зображення з високою оцінкою вважаються виявленнями. Для моделі YOLO використовується зовсім інший підхід. Єдина нейронна мережа застосовується до повного зображення. Ця мережа ділить зображення на регіони та передбачає обмежувальні рамки та ймовірності для кожного регіону. Ці обмежувальні рамки зважені за прогнозованими ймовірностями.

Модель YOLO має кілька переваг перед системами на основі класифікаторів. Вона розглядає все зображення під час тестування, тому її прогнози ґрунтуються на глобальному контексті зображення. Вона також робить прогнози за допомогою єдиної оцінки мережі, на відміну від таких систем, як R-CNN, які потребують тисячі ітерації для одного зображення. Це робить модель YOLO надзвичайно швидкою, більш ніж у 1000 разів швидшою, ніж R-CNN, і в 100 разів швидшою, ніж Fast R-CNN. Архітектура Yolo більше схожа на FCNN (повністю згорточна нейронна мережа) і передає зображення (nxn) один раз через FCNN, а виводом є передбачення (mxm). YOLO видає менше половини фонових помилок порівняно з Fast R-CNN. [8]

У табл. 1.1 містяться результати аналізу алгоритмів-аналогів розроблюваної системи.

Таблиця 1.1 – Аналіз аналогів розроблюваної системи

	Fast R-CNN	RetinalNet	R-FCN	YOLO	Розроблювана система
Відкритий код	+	-	+	-	+
Можливість інтеграції з іншими системами	+	-	-	+	+
Можливість підвищення точності	-	-	-	+	+
Низька вартість використання	+	-	+	-	+
Незалежність від устаткування	+	-	+	-	+

Незважаючи на те, що багато існуючих алгоритмів розпізнавання забезпечують досить високий рівень точності розпізнавання, розробка власної системи з

відкритим кодом та можливістю інтеграції з іншими системами є актуальним завданням.

1.5 Висновки до розділу

У розділі 1 розглянуто сучасний стан проблеми біометричної ідентифікації, включаючи проблеми конфіденційності. Прийнято рішення використання згорткової нейронної мережі у якості моделі машинного навчання при розробці системи ідентифікації. Розглянуто загальну модель нейронної мережі та архітектурні ідеї згорткових мереж.

Визначено базові метрики біометричної ідентифікації, що базуються на зображенні обличчя людини та на основі даних геометрії тіла.

Проведено аналіз аналогів та зроблено висновок щодо актуальності розробки.

2 РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ СИСТЕМИ

2.1 Оцінка ефективності систем біометричної ідентифікації

Оцінка ефективності системи біометричної ідентифікації є складною темою дослідження. Загальна продуктивність біометричної системи оцінюється з точки зору її точності, швидкості та зберігання даних. Кілька інших факторів, як-от вартість і простота використання, також впливають на ефективність.

Біометричні системи не є досконалими, і іноді помилково сприймають самозванця як дійсну особу (хибний збіг) або, навпаки, відхиляють дійсну особу (хибна невідповідність). Імовірність скоєння цих двох типів помилок називається коефіцієнтом помилкових невідповідностей (FNR) і коефіцієнтом помилкових збігів (FMR); величини цих помилок залежать від того, наскільки ліберально чи консервативно працює біометрична система. Компроміс між FMR і FNR системи в різних робочих точках називається «Робочими характеристики приймача (ROC - receiver operating characteristic)» і є комплексним показником точності системи в певному тестовому середовищі.

Програми доступу з високим рівнем безпеки, де велика занепокоєність щодо злому, працюють із малим FMR.

Криміналістичні застосування, де бажання зловити злочинця переважує незручність огляду великої кількості фальшиво звинувачених осіб, працюють зі своїм відповідником на високому FMR. Цивільні програми намагаються керувати своїми відповідниками в робочих точках як з низьким FNR, так і з низьким FMR. Частота помилок системи в робочій точці, де FMR дорівнює FNR, називається частотою рівних помилок (EER), яка часто може використовуватися як стислий опис точності системи.

Точність біометричної системи вважається прийнятною, якщо ризики (переваги), пов’язані з помилками в прийнятті рішень у даній робочій точці ROC для даного тестового середовища, є прийнятними. Подібним чином, точність

ідентифікації на основі біометричних даних є неприйнятною/низькою, якщо ризики (переваги), пов’язані з помилками, пов’язаними з будь-якою робочою точкою ROC для даного тестового середовища, є неприйнятними (недостатніми).

Розмір шаблону, кількість шаблонів, що зберігаються на кожного користувача, і доступність механізмів стиснення визначають обсяг пам’яті, необхідний для кожного користувача.

Якщо розмір шаблону великий і шаблони зберігаються в центральній базі даних, пропускна здатність мережі може стати вузьким місцем системи для ідентифікації. Типова смарт-карта може містити лише кілька кілобайт інформації (наприклад, 8 КБ), а в системах, які використовують смарт-карти для розподілу пам’яті шаблонів, розмір шаблону стає важливою проблемою дизайну.

Час, необхідний біометричній системі для прийняття рішення про ідентифікацію, є критичним для багатьох програм. Для типової програми керування доступом системі потрібно прийняти рішення про автентифікацію в режимі реального часу. У програмі банкомату, наприклад, бажано виконати автентифікацію протягом приблизно однієї секунди. Однак для судово-медичної експертизи вимоги до часу можуть бути не дуже жорсткими.

За інших незмінних факторів широке використання біометрії буде стимулюватись її впровадженням на споживчому ринку.

Найважливішим фактором, що впливає на цю реалізацію, є вартість біометричних систем, включаючи датчики та відповідну інфраструктуру. Деякі датчики, такі як мікрофони, вже дуже недорогі, тоді як інші, такі як CCD (Charged Coupled Device) камери, тепер стають стандартними периферійними пристроями в персональному комп’ютерному середовищі. Завдяки останнім досягненням твердотільних технологій датчики відбитків пальців стануть досить недорогими в найближчі кілька років. Вимоги до зберігання біометричних шаблонів і вимоги до обробки для відповідності є одними з двох основних міркувань щодо вартості інфраструктури.

2.2 Виявлення проблеми проведення біометричної ідентифікації

Для представлення предметної області застосовуються наступні методи: «дерево проблем», «дерево цілей», «риб'ячий скелет». За допомогою запропонованих методів виявлені проблема та актуальність теми магістерської комплексної теми кваліфікаційної роботи.

Дерево проблем орієнтовано на отримання відносно стійкої структури проблематики. Це дерево проблем дозволяє представити у компактному вигляді значний об’єм інформації щодо поточної проблеми. Також дерево проблем добре вирішує задачі виявлення та ранжування проблем, які є виявленими в досліджуваній темі.

Окремо слід зазначити суттєвий вклад дерева проблем у задачі класифікації, тобто розподіл проблем за відомими типами проблематики. Дерево проблем дозволяє наочно побачити співвідношення та взаємозв’язок різних типів проблематики за обраною темою дослідження. На рис. 2.1 наведено дерево проблем з тематики біометричної ідентифікації.

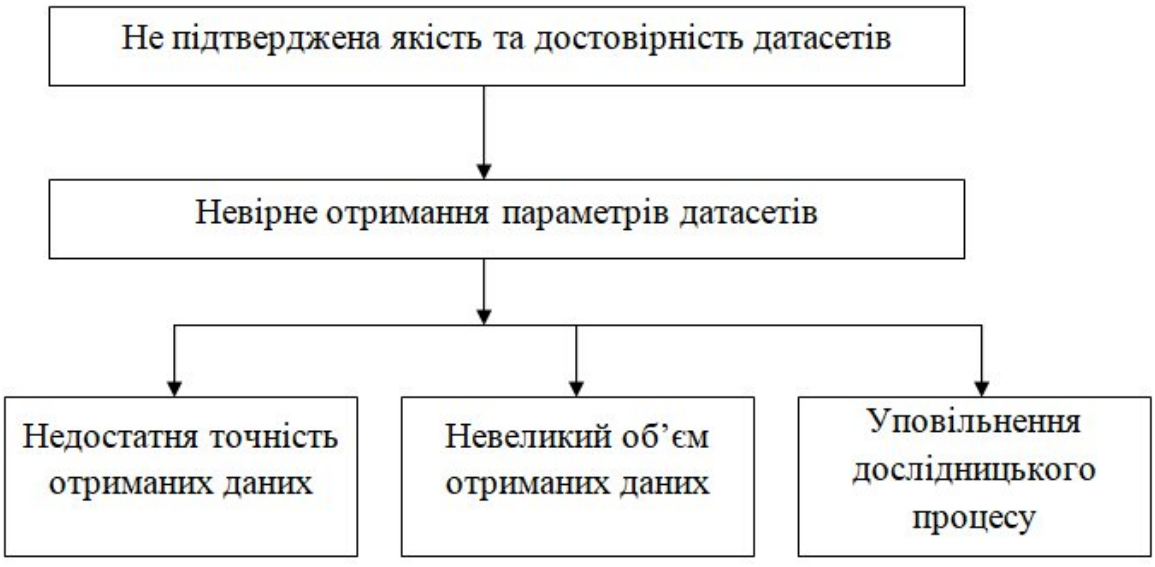

Рисунок 2.1 – Дерево проблем для теми біометричної ідентифікації

Загальна проблема біометричної ідентифікації зображення людини є в тому, що існуючі апаратні комплекси не в змозі з достатньої точністю автоматично проводити біометричну ідентифікацію. На рис. 2.2 наведена діаграма Ісікави для виявлення ключових взаємозв’язків між різними факторами та більш повного розуміння досліджуваного процесу. Ця діаграма також акцентується на проблемах предметної області, отже, дещо схожа з деревом проблем [9].

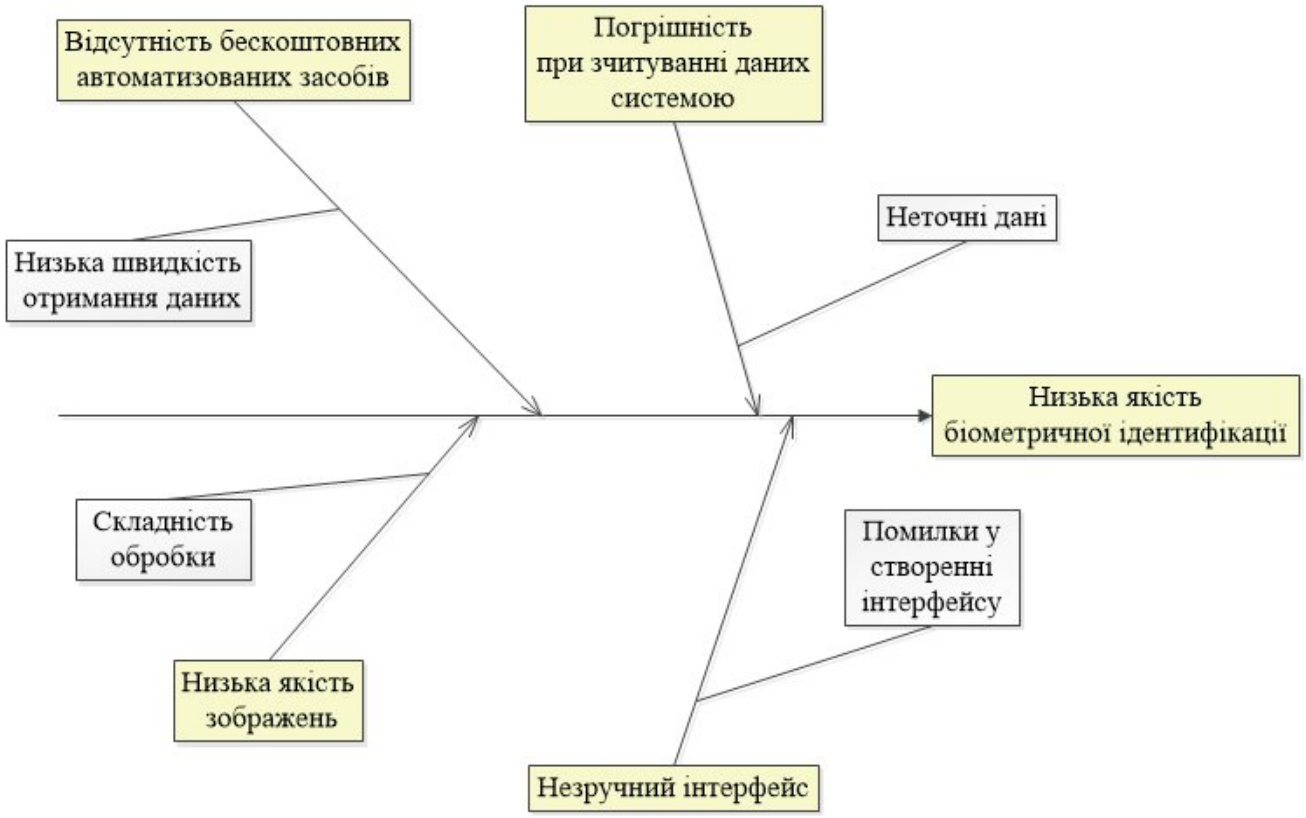

Рисунок 2.2 – Діаграма Ісікави досліджуваного процесу біометричної ідентифікації

Для вирішення означених проблем потрібно визначити цілі та зробити їх декомпозицію, отже, для більш детального та об’єктивного вивчення предметної області доцільно побудувати дерево цілей [10, 11].

Для досягнення цілей визначаються окремі задачі, процеси та ресурси, необхідні для здійснення процесів. Дерево цілей – це графічна схема, яка містить декомпозицію цілей на підцілі (рис. 2.3).

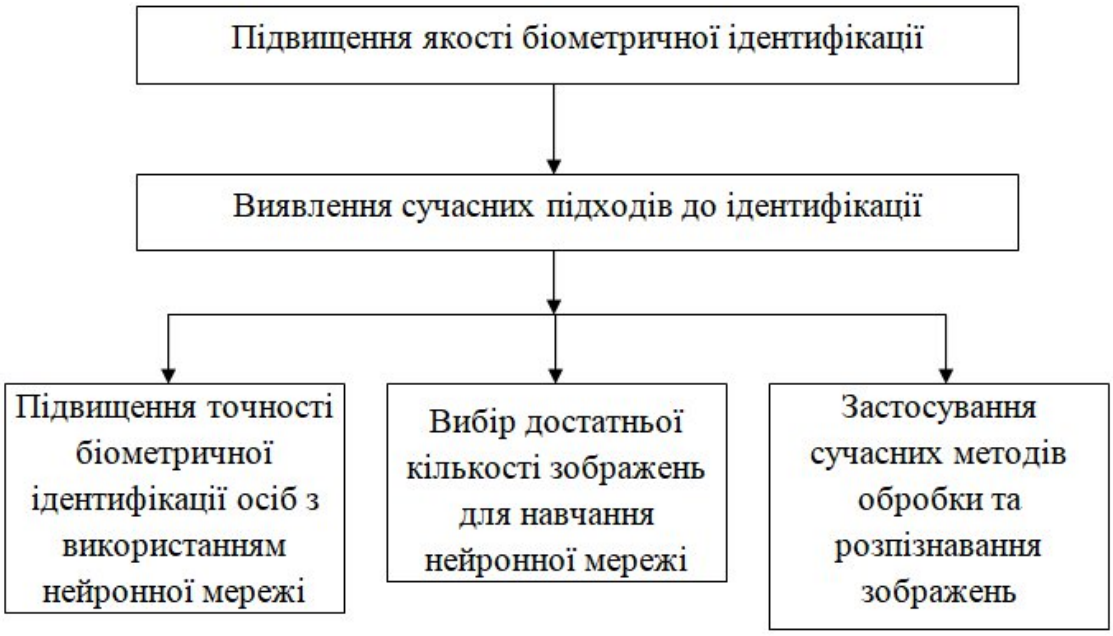

Рисунок 2.3 – Дерево цілей для підвищення якості біометричної ідентифікації

2.3 Застосування нейронної мережі для ідентифікації об’єктів

У файлах графічна інформація стосовно зображень зберігається відповідно до обраного графічного формату.

Нехай, відповідно до вимог, ставимо завдання проведенні біометричної ідентифікації на основі двовимірних зображень, що належать до наступних графічних форматів [12, 13]:

- Windows bitmaps — BMP, DIB;
- JPEG files — JPEG, JPG, JPE;
- Portable Network Graphics — PNG;
- TIFF files — TIFF, TIF.

Для того, щоб навчити нейронну мережу виявляти об'єкти на будь-якому зображенні, з приблизно однаковою формою і кольорами, слід застосовувати різні фільтри (рис. 2.4):

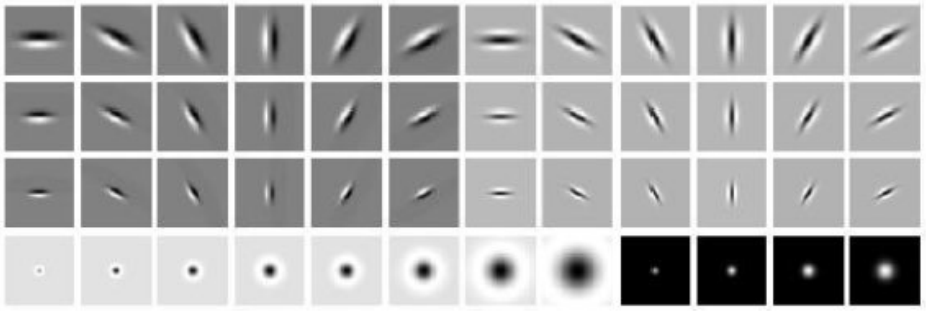

Рисунок 2.4 – Набір монохромних фільтрів для фрагментів зображень

За допомогою різних наведених фільтрів можна виділяти різноманітні фрагменти певного зображення. Потім їх можливо виявити, ідентифікувати та дослідити у вигляді окремих властивостей, а також передавати іншими шарами нейрона. У такому випадку для того, щоб мережам не доводилося окремо розпізнавати об'єкти в окремих частинах зображення, потрібно розділяти ваги, які відповідають за розпізнавання частин зображення, між різними частинами вихідного зображення.

Згортковий шар нейронної мережі є набором карт (карти ознак, features maps). Кожна карта має ядро, що сканує (скануюче ядро - являє собою фільтр, який ковзає по всьому зображенню і знаходить задані ознаки в будь-якому його місці). Кількість карток визначається вимогами до задачі, якщо взяти велику кількість карток, то підвищиться якість розпізнавання, але збільшиться обчислювальна складність.

Розмір у всіх карт згорткового шару однаковий і обчислюється за формулою:

$$(s, r) = (mS - kS + 1, mR - kR + 1) \quad (2.1)$$

де (s, r) – обчислюваний розмір згорткової карти, mS – ширина попередньої карти, mR – висота попередньої карти, kS – ширина ядра, kR – висота ядра.

Ядро ковзає по попередній карті і здійснює операцію згортка, яка часто використовується для обробки зображень:

$$(p*q)[m, n] = \Sigma p[m - k, n - l]\, k,l * q[k, l], \qquad (2.2)$$

де p - вихідна матриця зображення,

q - ядро згортки.

Процес відбувається наступним чином: вікном розміру ядра q проходимо із заданим кроком (зазвичай з кроком=1) все зображення p, кожному кроці поелементно множимо вміст вікна на ядро q, результат підсумовується і записується в матрицю результату (рис. 2.5). З матриць результату (карт згорткового шару) формується підсумкова карта ознак.

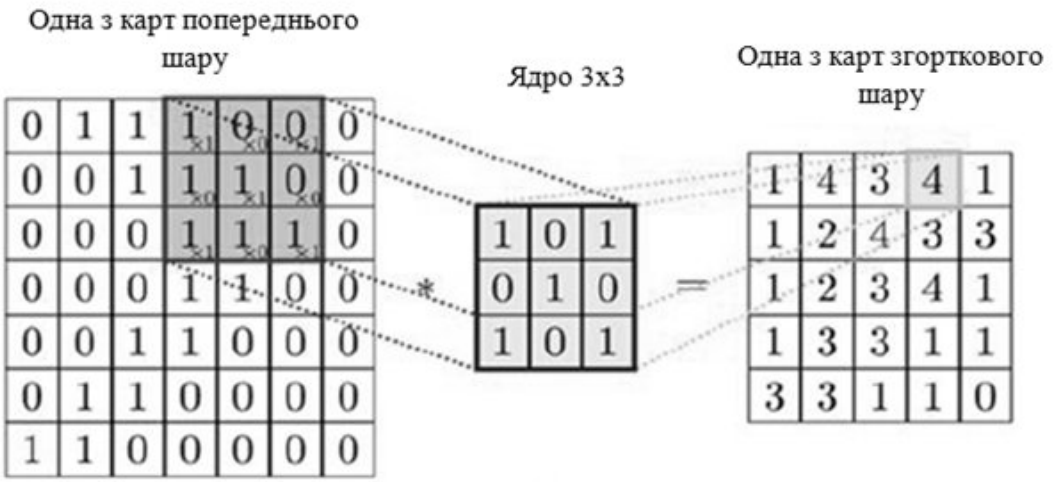

Рисунок 2.5 – Демонстрація операції згортки та отримання карти ознак

Підвибірковий шар також, як і згортковий, має карти, але їх кількість збігається з попереднім (згортковим) шаром. Ціль використання даного шару – зменшити розмірності карт попереднього шару. Якщо на попередній операції згортки вже були виявлені деякі ознаки, то для подальшої обробки настільки докладне зображення вже не потрібне і воно ущільнюється до менш докладного. До того ж, фільтрація вже непотрібних деталей допомагає не перевчитися.

У процесі сканування ядром підвибіркового шару (фільтром) карти попереднього шару, скануюче ядро не перетинається на відміну від згорткового шару. Зазвичай кожна карта має ядро розміром 2x2, що дозволяє зменшити попередні карти згорткового шару в 2 рази. Вся карта ознак поділяється на комірки 2x2 елементи, у тому числі вибираються максимальні за значенням. Зазвичай, у підвиборному шарі застосовується функція активації (ReLU, Rectified linear unit). Операція підвиборки виконується (Max-Pool – вибір максимального) відповідно до рис. 2.6.

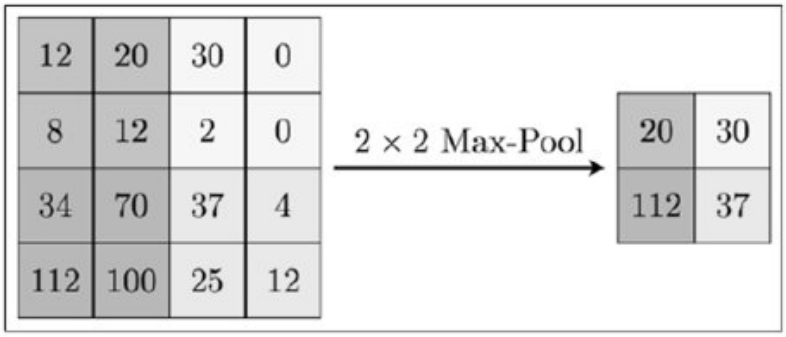

Рисунок 2.6 – Формування нової карти підвиборного шару на основі попередньої карти згорткового шару

Цей шар може бути описаний формулою:

$$x^l = f(a^l \cdot localmax(x^{l-1}) + b^l), \qquad (2.3)$$

де x^l - вихід шару l, f - функція активації, a^l, b^l – коефіцієнти зсуву шару l, *localmax* - операція вибірки локальних максимальних значень.

Останній тип шарів – це шар звичайного багатошарового персептрона. Мета застосування даного шару - це звернення до виходу попереднього шару та визначення властивостей, які пов'язані з певним класом.

Нейрони кожної карти попереднього підвиборчого шару пов'язані з одним прихованим нейроном шару. Таким чином число нейронів прихованого шару дорівнює числу карт підвибіркового шару, але зв'язки можуть бути не обов'язково такими, наприклад, тільки частина нейронів будь-якої з карт підвибіркового шару пов'язана з першим нейроном прихованого шару, а частина, що залишилася з другим, або всі нейрони першої карти пов'язані з нейронами 1 та 2 прихованого шару.

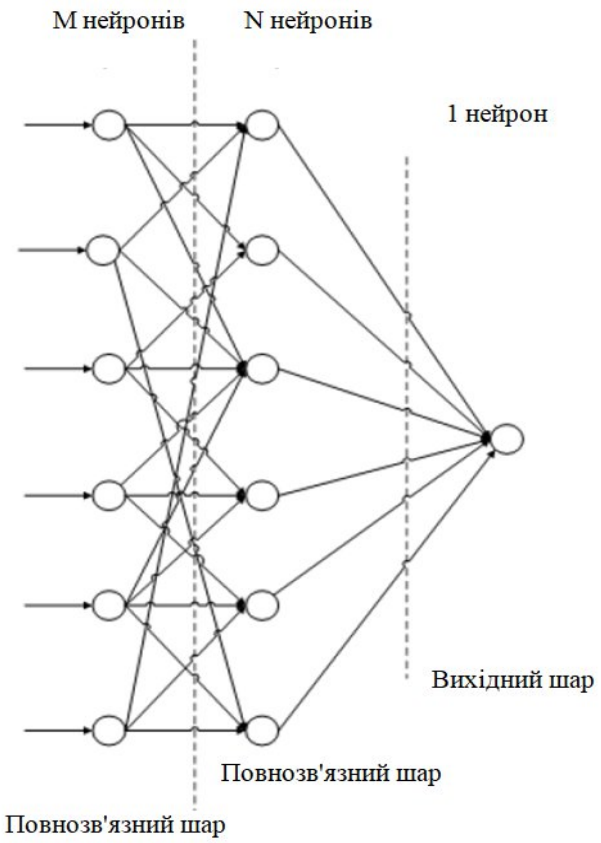

Рисунок 2.7 – Узагальнений вигляд повнозв’язаного шару нейромережі

На рис. 2.8 та 2.9 показані узагальнені нейромережі, що використовуються у даній роботі для біометричної ідентифікації.

Шар нейромережі містить фактично два екземпляри згорткової нейронної мережі. У першому варіанті (рис. 2.8) ця мережа натренована на датасеті зображень

обличчя, у другому варіанті (рис. 2.9) – на даних геометрії тіла людини. У разі, коли на вхід мережі потрапляють зображення однієї і тієї ж самої людини, результати X та Y повинні максимально наближатись один до одного.

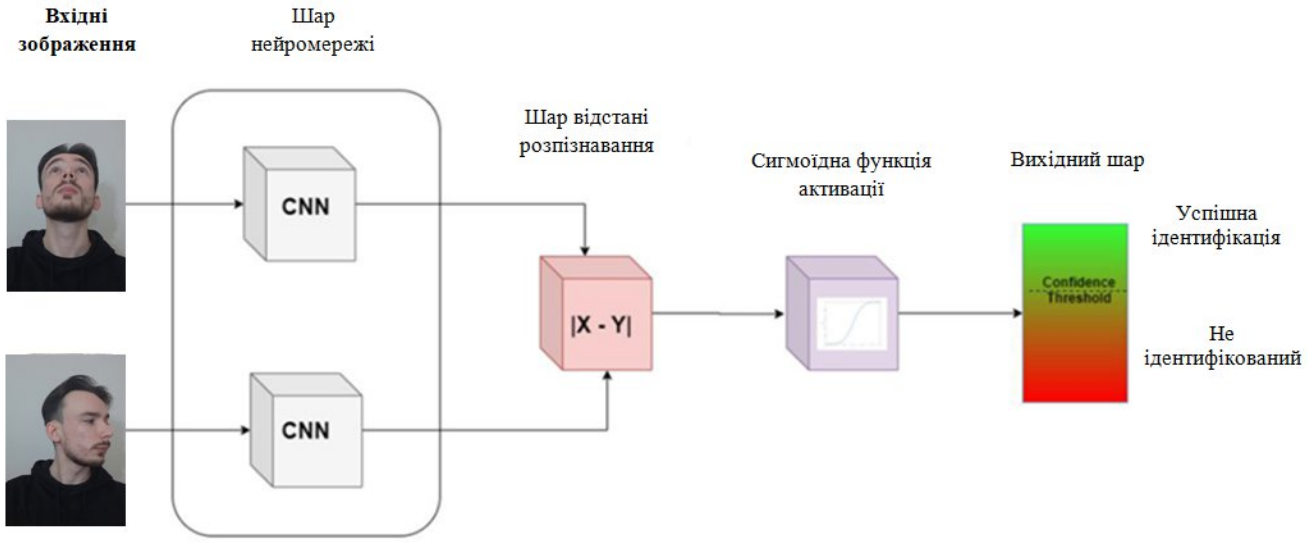

Рисунок 2.8 – Узагальнена нейронна мережа для біометричної ідентифікації за зображеннями обличчя людини

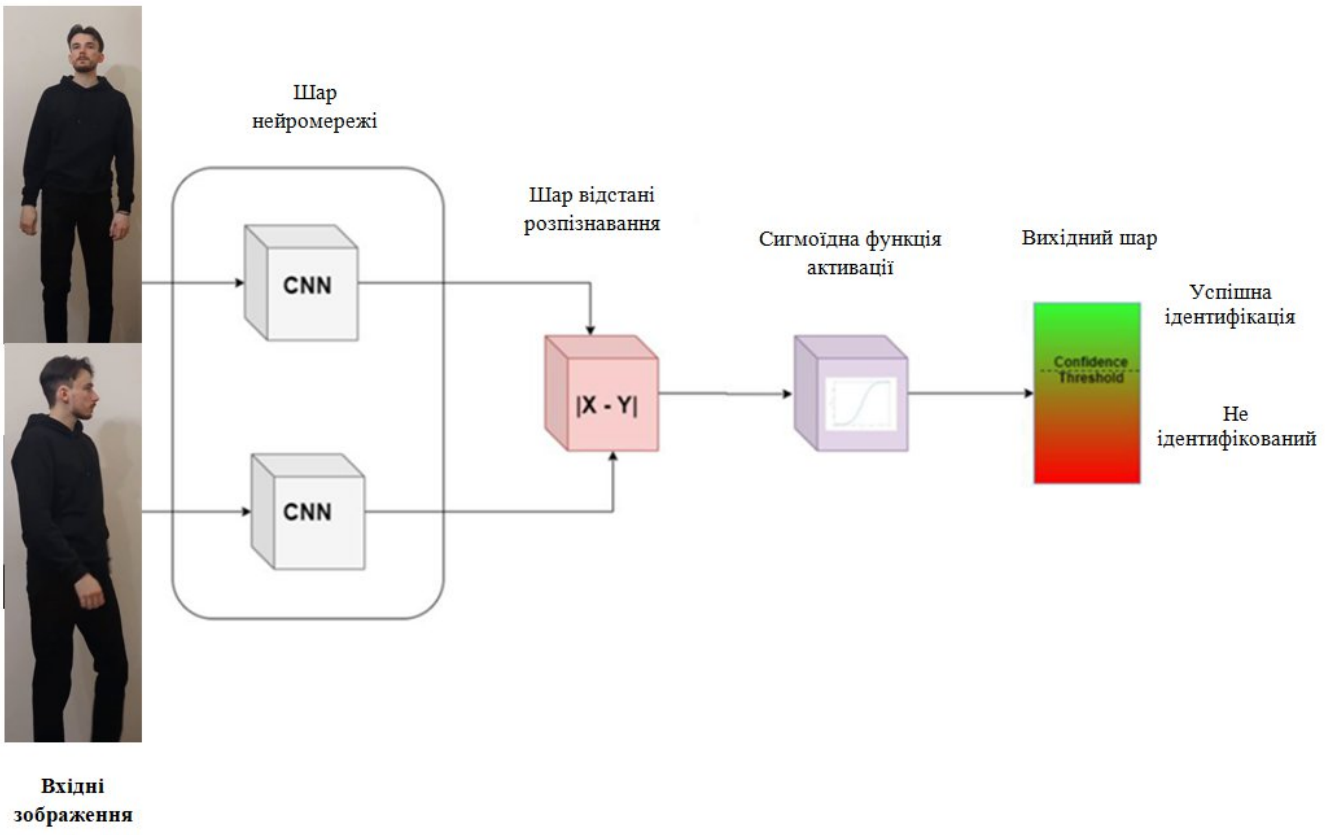

Рисунок 2.9 – Узагальнена нейронна мережа для біометричної ідентифікації за соматотипом людини

Таким чином, коли різниця |X-Y| наближається до 0, людина повинна бути ідентифікованою, а у разі великої різниці – ні.

На рис. 2.10 наведений екземпляр згорткової нейронної мережі. На рис. 2.8 та 2.9 такий екземпляр позначений написом CNN. Архітектура загорткової мережі обрана експериментальним шляхом.

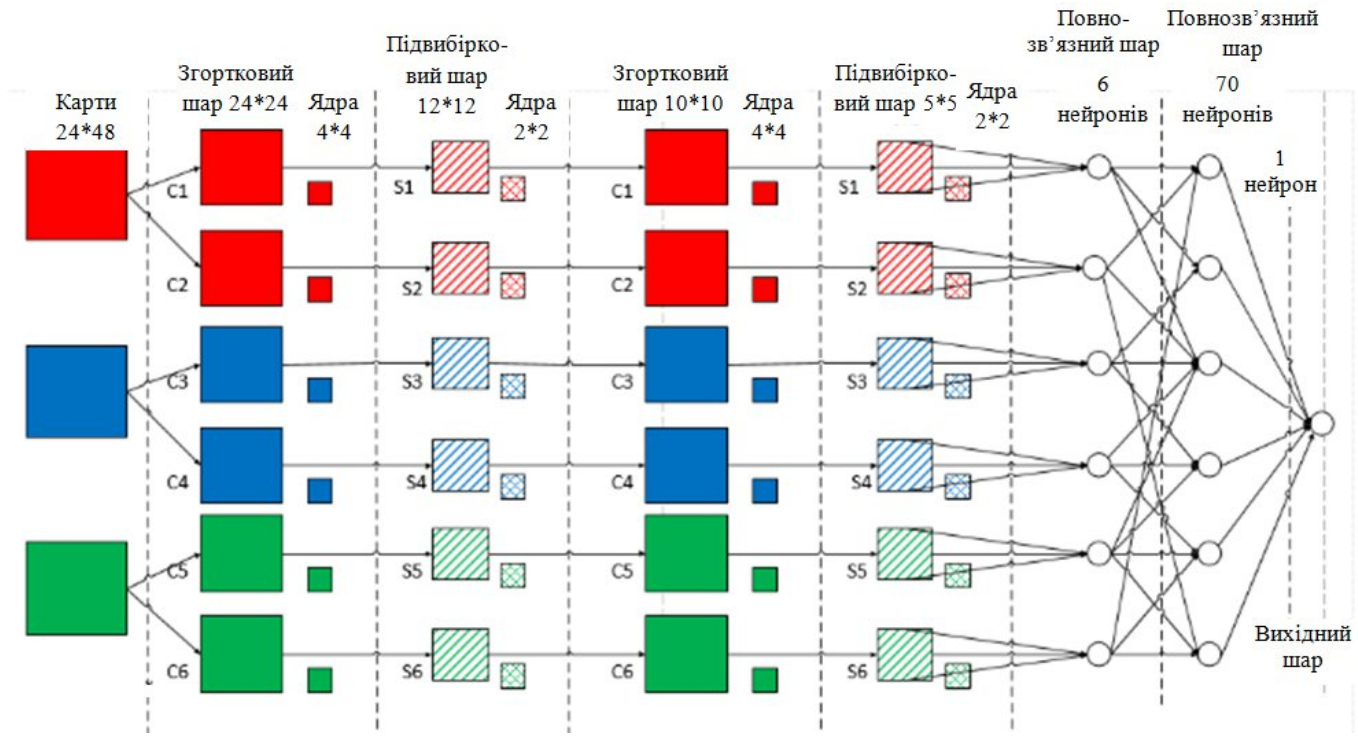

Рисунок 2.10 – Екземпляр згорткової нейронної мережі

На вхід мережі поступають RGB карти розміром 24x48. Далі йде згортковий шар, містить карти 24x24 та ядра 4x4. Підвибірковий шар містить карти 12x12 та ядра 2x2. Наступний згортковий шар містить карти 10x10 та ядра 4x4. Після цього йде підвибірковий шар з картами карти 5x5 та ядрами 2x2. За ним йдуть два повнозв'язні шари та вихідний шар.

Зазначимо, що топологія нейронної мережі (кількість згорткових та підвибіркових шарів, розміри карт та ядер) була обрана відповідно до загальних рекомендацій щодо топологій мереж для розпізнавання зображень, а також за результатами проведення експериментів, що надавали найвищу точність розпізнавання. Результати експериментів наведені у розділі 6.

2.4 Висновки до розділу

У розділі 2 розглянуті оцінки ефективності систем біометричної ідентифікації. Виявлено проблеми проведення біометричної ідентифікації. Створено дерево проблем для теми біометричної ідентифікації та діаграму Ісікави досліджуваного процесу біометричної ідентифікації. Для вирішення означених проблем розроблено дерево цілей, яке слугує для підвищення якості біометричної ідентифікації.

Оглянуто графічні формати, визначені формати, з якими повинна працювати програмна система: Windows bitmaps, JPEG files, Portable Network Graphics та TIFF files.

Проведено огляд шарів згорткової нейронної мережі, продемонстрована операція згортки та отримання карти ознак, а також процес формування нової карти підвиборного шару на основі попередньої карти згорткового шару.

Визначена топологія узагальненої нейронної мережі для біометричної ідентифікації людини за зображеннями обличчя та за соматотипом. Створений екземпляр згорткової нейронної мережі для RGB-зображень.

3 ВИМОГИ ДО СЕРВЕРНОЇ ЧАСТИНИ СИСТЕМИ БІОМЕТРИЧНОЇ ІДЕНТИФІКАЦІЇ

3.1 Опис вимог до серверної частини системи з описом прецедентів

Система машинного зору, призначена для біометричної ідентифікації людини за зображенням обличчя чи геометрії тіла, містить розвинутий набір функцій. Для реалізації цих функцій повинна бути задіяна як серверна, так і клієнтська складові. Тільки функція оцінки якості біометричної класифікації має лише серверну складову. Для наочності специфікації функціональних вимог створено UML діаграму використання розроблюваної системи машинного зору (рис. 3.1).

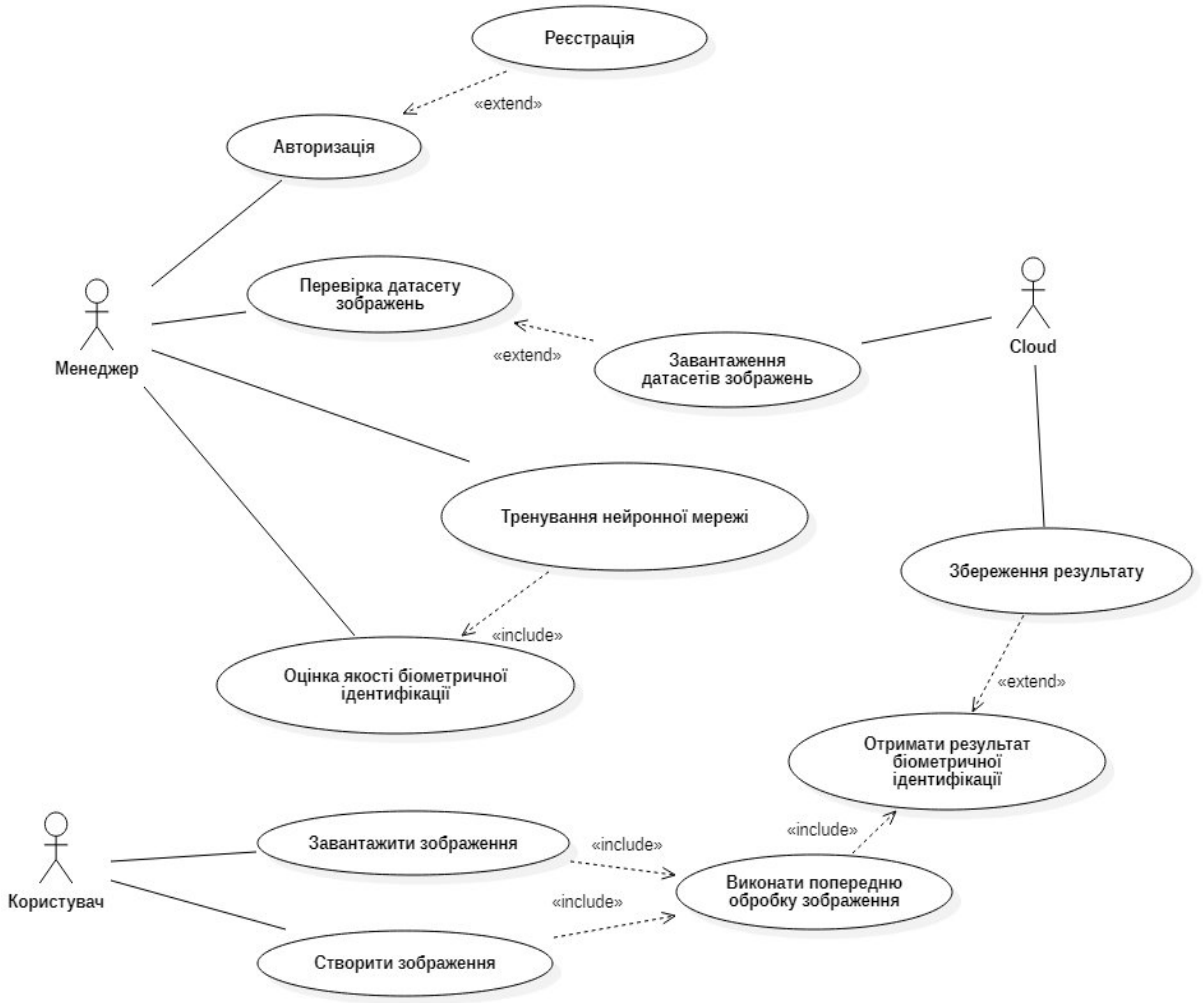

Рисунок 3.1 – Use Case серверної частини системи біометричної ідентифікації

У наступних таблицях показані прецеденти та специфіка серверної частини (СЧ) функціоналу, яка взаємодіє з клієнтською частиною (КЧ).

Таблиця 3.1 – Опис варіанта використання «Реєстрація»

Характеристики ВВ	Опис ВВ (СЧ)
Назва	Реєстрація
Порядковий номер ВВ	1
Короткий опис	ВВ дозволяє зареєструвати користувача системи машинного зору для користування системою.
Актори	Будь-який користувач
Передумови	1. Неавторизований користувач 2. Користувач бажає авторизуватись як менеджер
Основний потік	1. СЧ отримує повідомлення від КЧ, що натиснута кнопка реєстрації, та пару логін+пароль. 2. СЧ перевіряє формат даних. 3. СЧ звертається до репозиторію та перевіряє унікальність логіну. 4. СЧ створює новий акаунт. 5. СЧ передає повідомлення КЧ про успіх реєстрації. 6. СЧ надає менеджеру доступ до функціоналу.
Постумови	1. Користувач зареєстрований як менеджер. 2. Профіль нового менеджера збережений у БД.
Альтернативний потік № 1	1.Користувач ввів дані не повністю (заповнив не всі поля). 2. СЧ повертає на КЧ помилку «не заповнені поля».

Продовження таблиці 3.1

Характеристики ВВ	Опис ВВ
Альтернативний потік № 2	1. Користувач ввів дані помилкового формату. 2. СЧ повертає на КЧ помилку «помилка формату».
Альтернативний потік № 3	1. Введений логін належить іншому користувачу. 2. СЧ повертає на КЧ помилку «пароль зайнятий».

Таблиця 3.2 – Опис варіанта використання «Авторизація»

Характеристики ВВ	Опис ВВ (СЧ)
Назва	Авторизація
Порядковий номер ВВ	2
Короткий опис	ВВ дозволяє авторизувати менеджера.
Актори	Неавторизований Manager
Передумови	1.Неавторизований менеджер має профіль у системі. 2. Користувач бажає авторизуватись як менеджер.
Основний потік	1. СЧ отримує повідомлення від КЧ, що натиснута кнопка авторизації, та пару логін+пароль. 2. СЧ перевіряє формат даних. 3. СЧ звертається до репозиторію та перевіряє наявність пари логін+пароль. 4. СЧ передає повідомлення КЧ про успіх авторизації. 5. СЧ надає менеджеру доступ до функціоналу.
Постумови	1. Користувач авторизований як менеджер. 2. Застосунок підтверджує успіх авторизації. Менеджер може користуватись системою.

Продовження таблиці 3.2

Характеристики ВВ	Опис ВВ
Альтернативний потік № 1	1.Користувач ввів дані не повністю (заповнив не всі поля). 2. Менеджер не авторизований.
Альтернативний потік № 2	1. Користувач ввів дані помилкового формату. 2. СЧ повертає на КЧ помилку «помилка формату». Потрібно повторення вводу.

Таблиця 3.3 – Опис варіанта використання «Завантаження датасетів зображень»

Характеристики ВВ	Опис ВВ (СЧ)
Назва	Завантаження датасетів зображень
Порядковий номер ВВ	3
Короткий опис	ВВ дозволяє завантажити потрібний датасет зображень.
Актори	Manager
Передумови	1. Менеджер авторизований у системі. 2. На сервері розміщений потрібний датасет.
Основний потік	1. СЧ отримує запит від КЧ на отримання переліку датасетів. 2. СЧ повертає на КЧ список датасетів та їх опис (кількість зображень, тип зображень тощо). 3. СЧ отримує запит від КЧ на завантаження визначеного датасету. 4. СЧ повертає на КЧ посилання на зображення з датасету.

Продовження таблиці 3.3

Характеристики ВВ	Опис ВВ
Постумови	1. Користувач отримав зображення та може їх переглядати. 2. Обраний користувачем датасет вважається поточним
Альтернативний потік № 1	1. Список датасетів недоступний. 2. СЧ повертає на КЧ помилку «помилка серверу».
Альтернативний потік № 2	1. Зображення, що входять у датасет, недоступні. 2. СЧ повертає на КЧ помилку «помилка завантаження датасету».

Таблиця 3.4 – Опис варіанта використання «Перевірка датасету зображень»

Характеристики ВВ	Опис ВВ (СЧ)
Назва	Перевірка датасету зображень
Порядковий номер ВВ	4
Короткий опис	ВВ дозволяє переглядати зображення з обраного датасету.
Актори	Manager
Передумови	1. Менеджер обрав датасет. 2. КЧ отримала посилання на зображення.
Основний потік	1. СЧ отримує запит від КЧ на відображення порції зображень. 2. СЧ перевіряє цілісність обраних зображень 3. СЧ надає КЧ зображення за посиланням.

Продовження таблиці 3.4

Характеристики ВВ	Опис ВВ
Постумови	1. Користувач переглянув зображення.
Альтернативний потік № 1	1. Проблеми інтернет-зв’язку. 2. СЧ повертає на КЧ помилку «перевірте інтернет-з’єднання».

Таблиця 3.5 – Опис варіанта використання «Тренування нейронної мережі»

Характеристики ВВ	Опис ВВ (СЧ)
Назва	Тренування нейронної мережі
Порядковий номер ВВ	5
Короткий опис	ВВ дозволяє натренувати нейронну мережу на основі датасету.
Актори	Manager
Передумови	1. Менеджер обрав датасет.
Основний потік	1. СЧ отримує запит від КЧ на тренування нейронної мережі. 2. СЧ запускає процес тренування. 3. СЧ надає КЧ повідомлення про завершення тренування.
Постумови	1. Нейронна мережа натренована.
Альтернативний потік № 1	1. Проблеми інтернет-зв’язку. 2. СЧ повертає на КЧ помилку «перевірте інтернет-з’єднання».

Таблиця 3.6 – Опис варіанта використання «Оцінка якості біометричної ідентифікації»

Характеристики ВВ	Опис ВВ (СЧ)
Назва	Оцінка якості біометричної ідентифікації
Порядковий номер ВВ	6
Короткий опис	ВВ дозволяє провести тестування натренованої нейронної мережі та отримати оцінку точності
Актори	Manager
Передумови	1. Нейронна мережа натренована. Менеджер бажає отримати оцінки точності мережі.
Основний потік	1. СЧ обирає частину зображень, призначену для тестування натренованої нейронної мережі. 2. СЧ перевіряє цілісність обраних зображень 3. СЧ виконує тестування нейронної мережі. 4. СЧ розраховує оцінку точності. 5. СЧ передає на КЧ оцінку для можливості перегляду її менеджером.
Постумови	1. Менеджер отримав оцінки точності.
Альтернативний потік № 1	1. Зображення, що входять у частину зображень, призначену для тестування, недоступні. 2. СЧ повертає на КЧ помилку «помилка завантаження датасету».
Альтернативний потік № 2	1. Проблеми інтернет-зв’язку. 2. СЧ повертає на КЧ помилку «перевірте інтернет-з’єднання».

Таблиця 3.7 – Опис варіанта використання «Завантажити зображення»

Характеристики ВВ	Опис ВВ (СЧ)
Назва	Завантажити зображення
Порядковий номер ВВ	7
Короткий опис	ВВ дозволяє завантажити окреме зображення за вимогою користувача
Актори	Будь-який користувач
Передумови	1. Користувач бажає завантажити зображення.
Основний потік	1. СЧ отримала від КЧ запит на обробку зображення. 2. СЧ запускає варіант використання «Виконати попередню обробку зображення».
Постумови	1. Користувач завантажив зображення.
Альтернативний потік № 1	1. Зображення, що обрано для завантаження, пошкоджено. 2. СЧ повертає на КЧ помилку «помилка завантаження зображення».
Альтернативний потік № 2	1. Проблеми інтернет-зв’язку. 2. СЧ повертає на КЧ помилку «перевірте інтернет-з’єднання».

Таблиця 3.8 – Опис варіанта використання «Створити зображення»

Характеристики ВВ	Опис ВВ (СЧ)
Назва	Створити зображення
Порядковий номер ВВ	8

Продовження таблиці 3.8

Характеристики ВВ	Опис ВВ
Короткий опис	ВВ дозволяє створити зображення користувача
Актори	Будь-який користувач
Передумови	1. Користувач бажає створити зображення.
Основний потік	1. СЧ звертається до об’єкта Camera Android для отримання зображення користувача. 2. СЧ отримала від КЧ запит на обробку зображення. 3. СЧ запускає варіант використання «Виконати попередню обробку зображення».
Постумови	1. Користувач створив зображення.
Альтернативний потік № 1	1. Доступ до камери неможливий. 2. СЧ повертає на КЧ помилку «помилка доступу до камери».
Альтернативний потік № 2	1. Проблеми інтернет-зв’язку. 2. СЧ повертає на КЧ помилку «перевірте інтернет-з’єднання».

Таблиця 3.9 – Опис варіанта використання «Виконати попередню обробку зображення»

Характеристики ВВ	Опис ВВ (СЧ)
Назва	Виконати попередню обробку зображення
Порядковий номер ВВ	9
Короткий опис	ВВ дозволяє створити обробити зображення, яке було завантажено користувачем або зроблено на камеру

Продовження таблиці 3.9

Характеристики ВВ	Опис ВВ
Актори	Будь-який користувач
Передумови	1. Користувач бажає покращити якість зображення.
Основний потік	1. СЧ отримує зображення користувача. 2. СЧ застосовує алгоритм «Перетворення рівня яскравості». 3. СЧ застосовує алгоритм «Зниження шуму». 4. СЧ застосовує алгоритм «Розрахунок градієнта». 5. СЧ застосовує алгоритм «Детектор кутів Харріса». 6. СЧ запускає варіант використання «Отримати результат біометричної ідентифікації».
Постумови	1. Користувач отримав зображення з підвищеною якістю.
Альтернативний потік № 1	1. Проблеми інтернет-зв’язку. 2. СЧ повертає на КЧ помилку «перевірте інтернет-з’єднання».

Таблиця 3.10 – Опис варіанта використання «Отримати результат біометричної ідентифікації»

Характеристики ВВ	Опис ВВ (СЧ)
Назва	Отримати результат біометричної ідентифікації
Порядковий номер ВВ	10
Короткий опис	ВВ дозволяє отримати результат біометричної ідентифікації на основі двовимірного зображення

Продовження таблиці 3.10

Характеристики ВВ	Опис ВВ
Актори	Будь-який користувач
Передумови	1. Користувач бажає отримати результат ідентифікації.
Основний потік	1. СЧ виконує ідентифікацію на обробленому зображення користувача. 2. СЧ передає результати ідентифікації на КЧ.
Постумови	1. Користувач отримав результати ідентифікації та значення точності.
Альтернативний потік № 1	1. Проблеми інтернет-зв’язку. 2. СЧ повертає на КЧ помилку «перевірте інтернет-з’єднання».

Таблиця 3.11 – Опис варіанта використання «Збереження результату»

Характеристики ВВ	Опис ВВ (СЧ)
Назва	Збереження результату
Порядковий номер ВВ	11
Короткий опис	ВВ дозволяє зберегти результат в хмарному середовищі
Актори	Будь-який користувач
Передумови	1. Користувач має акаунт та авторизований в хмарному середовищі. 2. Користувач отримав результат біометричної ідентифікації.

Продовження таблиці 3.11

Характеристики ВВ	Опис ВВ
Основний потік	1. СЧ отримує від СЧ повідомлення про необхідність збереження результатів. 2. СЧ зберігає дані. 3. СЧ надсилає повідомлення про успіх збереження на КЧ.
Постумови	1. Дані є збереженими. Користувач отримав відповідне повідомлення.
Альтернативний потік № 1	1. Користувач не має акаунту. 2. Результати зберігаються на пристрої.
Альтернативний потік № 2	1. Проблеми інтернет-зв’язку. 2. СЧ повертає на КЧ помилку «перевірте інтернет-з’єднання».

3.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якість системи за певними критеріями.

Під час формування нефункціональних вимог були визначені сценарії якості для різних атрибутів якості.

Для обраних варіантів використання необхідно визначити певні нефункціональні вимоги для частини функціоналу, що оглядається у даній роботі.

Атрибут «Безпека»:

«Сценарій безпеки: доступ до датасету»:

джерело – авторизований менеджер;

вплив (стимул) - спроба перевірити датасет зображень;

артефакт - сховище даних;

середовище – нормальне;

реакція - надання доступу та запуск перевірки;

міра реакції - менеджер отримує доступ до даних у датасеті більш ніж в 98,5% випадків.

«Сценарій для отримання результатів біометричної ідентифікації»

джерело – користувач;

вплив (стимул) – користувач завантажив чи створив двовимірне зображення;

середовище – нормальне;

реакція – користувач отримує результати ідентифікації з можливістю їх подальшого збереження;

міра реакції - користувач отримує результати на екрані пристрою не менше ніж у 98% випадків та може їх зберегти не менше ніж у 96,5% випадків на Cloud та не менше ніж у 97% випадків на власному пристрої.

Атрибут «Зручність використання»:

«Сценарій юзабіліті для тренування нейромережі»:

джерело – менеджер;

вплив (стимул) – запуск на навчання нейромережі;

реакція – менеджер обрав датасет та запустив тренування мережі;

міра реакції – для запуску тренування знадобилось 1 натискання кнопки.

«Сценарій юзабіліті для авторизації менеджера»:

джерело – менеджер;

вплив (стимул) – робить спробу зайти у систему;

реакція – менеджер перейшов на форму авторизації;

міра реакції – менеджер авторизувався завдяки заповненню 2 полів та 1 натисканню кнопки для підтвердження вводу.

«Сценарій юзабіліті для створення зображення»

джерело – користувач;

вплив (стимул) – активація камери;

реакція - користувач створив зображення на камеру;

міра реакції - користувач виконав дії в 3 натискання кнопок.

Атрибут «Продуктивність»:

«Сценарій продуктивності помірної кількості зображень у датасеті»

джерело – система;

вплив (стимул) – помірна (1000-10000) кількість зображень для навчання нейронної мережі;

артефакт – процес;

середовище - нормальне;

реакція - обробка всіх зображень;

міра реакції –обробка датасету повинна зайняти не більш ніж 3 секунди.

«Сценарій продуктивності великої кількості зображень у датасеті»

джерело – система;

вплив (стимул) – занадто велика кількість зображень для навчання нейронної мережі;

артефакт – процес;

середовище - режим підвищеної завантаженості;

реакція - обробка всіх зображень;

міра реакції – якщо обробка датасету займає більш ніж 3 секунди, потрібно надавати користувачеві progress bar.

Атрибут «Готовність»:

«Сценарій готовності для визначення попередньої обробки двовимірного зображення»:

джерело – система;

вплив (стимул) – погіршена якість зображення;

артефакт - підсистема обробки зображень;

середовище - погіршений режим;

реакція – застосування алгоритмів покращення якості зображення;

міра реакції – дія виконується у 99% випадків.

«Сценарій готовності повідомлення менеджеру про авторизацію»:

джерело – система;

вплив (стимул) – спроба авторизації;

артефакт – процес;

середовище – нормальне;

реакція – надсилання авторизаційних даних на сервер для пошуку аккаунту;

міра реакції - успіх не менше ніж в 98% випадків.

3.3 Висновки до розділу

У даному розділі визначено функціональні та нефункціональні вимоги до серверної частини розроблюваної системи.

Створено діаграму UseCase, описана взаємодія серверної складової з клієнтською.

Для формулювання нефункціональних вимог розглянуто атрибути безпеки, зручності використання, продуктивності та готовності.

4 ПРОЕКТУВАННЯ СЕРВЕРНОЇ ЧАСТИНИ СИСТЕМИ

4.1 Проектування архітектури системи

«Клієнт-серверну архітектуру можна означити, як концепцію інформаційної мережі в якій основна частина її ресурсів зосереджена в серверах, обслуговуючих своїх клієнтів. Така архітектура визначає такі типи компонентів

- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.» [14]

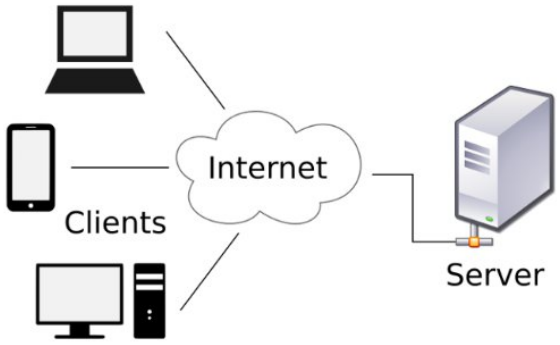

Рисунок 4.1 – Загальна схема клієнт-серверної архітектури

Для зручного використання системи було вирішено розділити систему на клієнтську і серверну. Тому для того, щоб зробити систему більш зручною і масштабованою, було прийнято рішення використовувати клієнт-серверну архітектуру, переваги якої перечисленні нижче [15].

- Оскільки всі дані зберігаються на сервері, легко зробити резервну копію.
- Правильне управління - всі важливі дані зберігаються в одному місці.
- Мережа клієнт-сервер має централізоване управління. тобто централізовані облікові записи користувачів, безпеку і доступ для спрощення адміністрування мережі.

- Зменшується реплікація даних - дані зберігаються на серверах замість кожного клієнта.
- Систему можна масштабувати для підтримки великої кількості клієнтів одночасно.

Клієнт-серверна архітектура має недоліки. Вони перераховані нижче.

- Потрібні професійні IT-фахівці для обслуговування серверів та інших технічних деталей системи.
- Установка і управління є дорогим, так як потрібне спеціальне обладнання (сервер) і спеціальне забезпечення як для серверу, так і для клієнта.
- Відмова сервера призводить до відмови всієї системи.

UML є де-факто стандартною нотацією для графічного представлення програмного забезпечення. Діаграми UML використовуються для аналізу, побудови та обслуговування програмних систем. Здебільшого діаграми UML відображають абстрактний вигляд (частини) програмної системи. Основна мета діаграм UML – обмінюватися знаннями про систему між розробниками. Якість компонування діаграм UML відіграє вирішальну роль у їх розумінні.

Розглянемо концептуальну модель класів системи. На рис. 4.2 зображена діаграма концептуальних класів системи машинного зору, яка містить класи та інтерфейси. Розглянемо їх більш детально.

Клас AndroidUser – абстрактний користувач, від якого успадковуються користувачі User та Manager.

Клас User – призначений для роботи звичайного користувача системи.

Клас Manager – призначений для роботи користувача з розширеним функціоналом, який може тренувати та налаштовувати нейромережу.

User Interface – інтерфейс звичайного користувача системи.

Manager Interface – інтерфейс менеджера.

Клас NeuralNetwork – призначений для збереження методів та властивостей для роботи нейромережі.

Клас Train – призначений для проведення тренування нейромережі.

Клас FaceDatasetClass – призначений для забезпечення роботи з датасетом зображень обличчя.

Клас BodyGeometryClass – призначений для забезпечення роботи з датасетом повних зображень тіла людини.

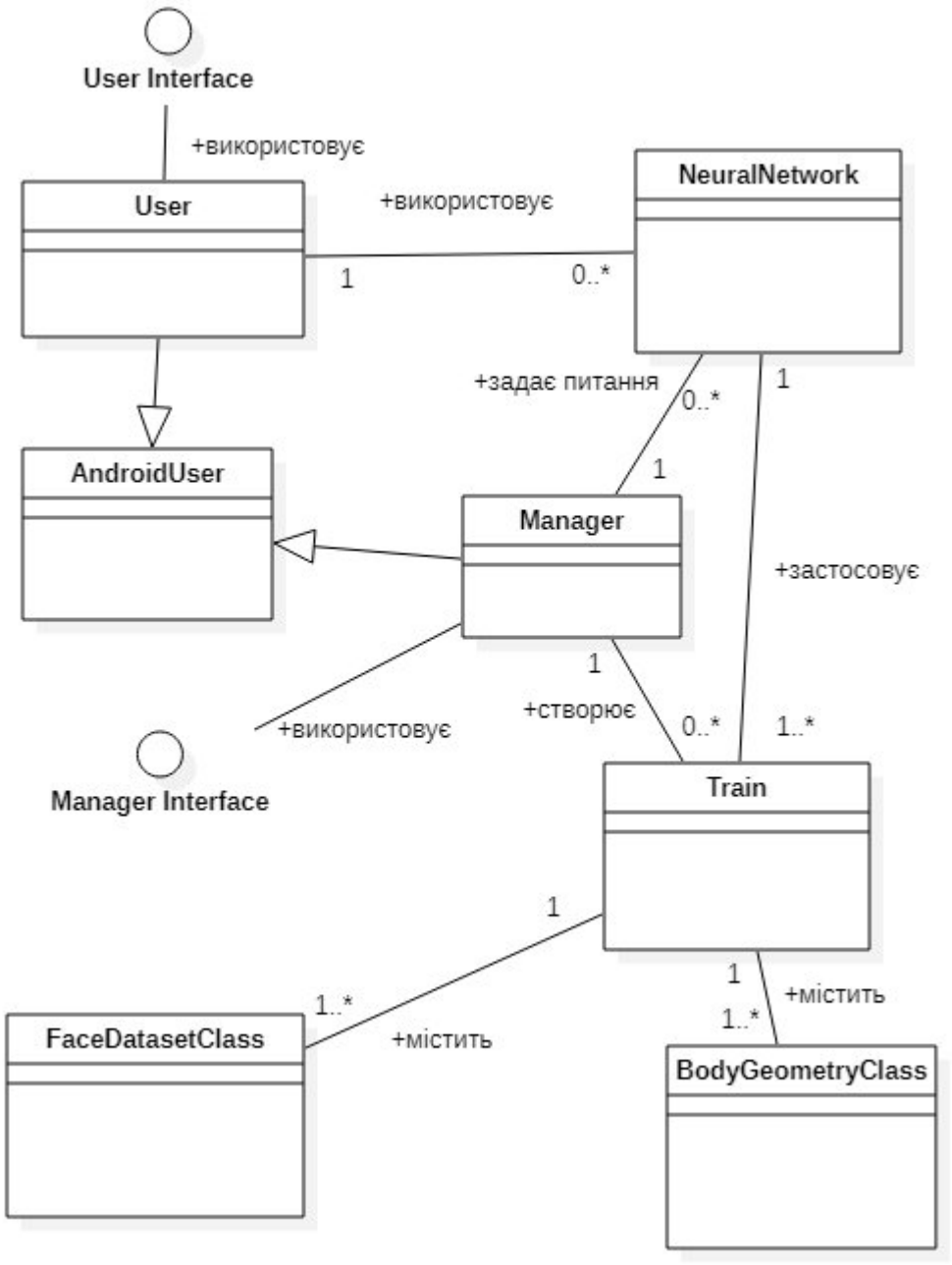

Рисунок 4.2 – Діаграма концептуальних класів системи машинного зору

Для подальшого проектування системи потрібно формалізувати основні процеси, які в ній відбуваються, та довизначити концептуальну діаграму конкретними методами та властивостями, що стосуються серверної частини системи машинного зору.

4.2 Діаграми діяльності роботи системи

Діаграми діяльності є широко поширеним засобом моделювання для представлення управління та потоку даних у програмних системах. Нотація застосовна до різних предметних областей та корисна на багатьох рівнях абстракції. Діаграми діяльності можна використовувати для низькорівневих описів алгоритмів, подібних до блок-схем, для моделювання об’єктів, що співпрацюють у системі на основі об’єктів, або для визначення простих потоків сторінок веб-застосунків і робочих процесів бізнес-продуктів високого рівня.

Основна ідея діаграм діяльності полягає в моделюванні дій і можливих порядків їх виконання. Окрім цього спільного знаменника, інтерпретація того, що являє собою дія та як визначити, коли та як дію активовано або коли вона завершує виконання, залишається специфічною для області застосування. Методичне призначення діаграм діяльності також підлягає інтерпретації щодо конкретного проекту: вона може вільно використовуватися для цілей документування або формально використовуватися для аналізу чи генерації коду.

Формальна семантика для діаграм діяльності допомагає зменшити непорозуміння між людьми та може покращити взаємодію між інструментами.

У даній роботі можна виділити два ключових процеси, а саме:

- тренування нейронної мережі з обраною топологією;
- виконання біометричної ідентифікації за зображеннями обличчя чи геометрії тіла людини.

На рис. 4.3 наведена діаграма діяльності, яка демонструє процес тренування нейронної мережі.

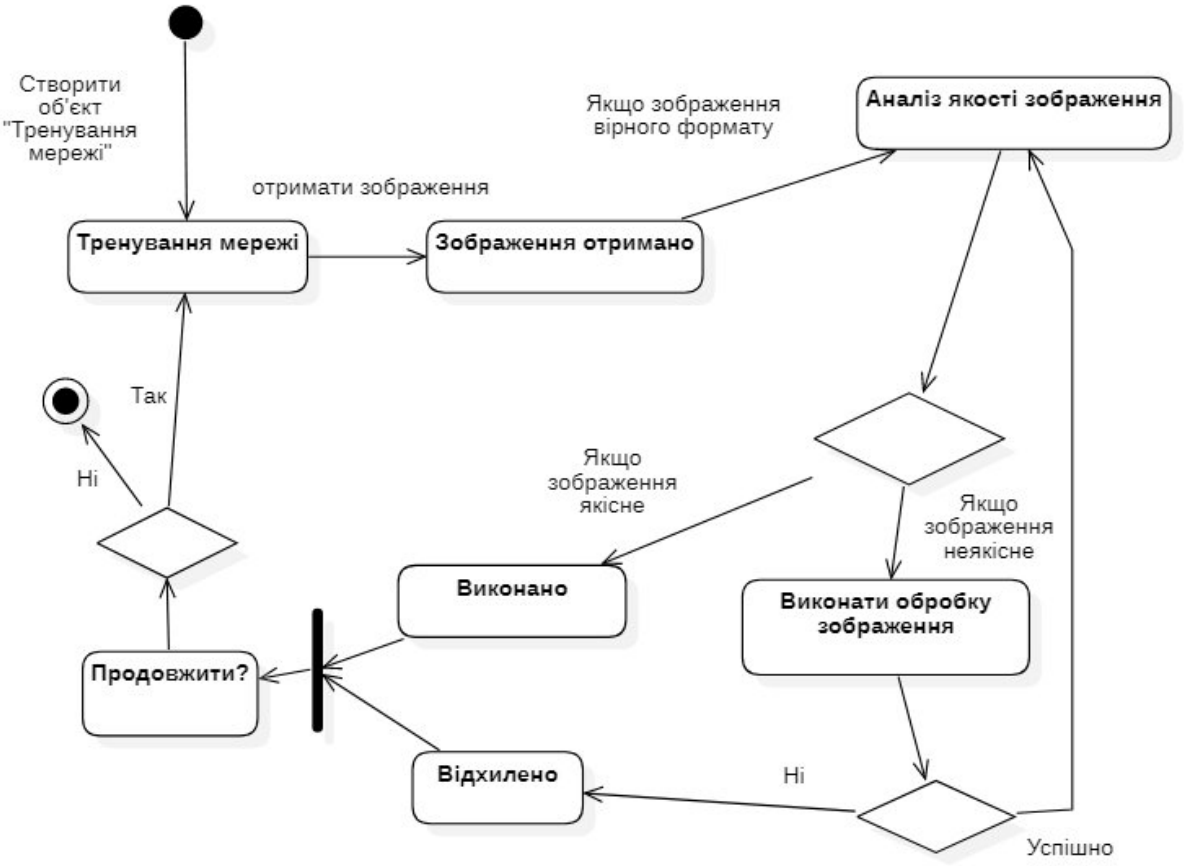

Рисунок 4.3 – Діаграма діяльності тренування нейронної мережі

Як можна побачити, для зображення, яке потенційно може увійти у навчальний датасет, спочатку перевіряється відповідність формату. Якщо файл не є графічним зображенням, або пошкоджений, він відкидається. Потім аналізується якість представленого зображення. Якщо перевірка показує, що якість може бути підвищена, застосовуються відповідні алгоритми («Перетворення рівня яскравості», «Зниження шуму», «Розрахунок градієнта», «Детектор кутів Харріса»). Після застосування алгоритму знову перевіряється якість зображення. Якщо зображення добре, воно використовується для тренування нейромережі.

На рис. 4.4 показана діаграма діяльності для процесу біометричної ідентифікації.

Рисунок 4.4 – Діаграма діяльності виконання біометричної ідентифікації

Після визначення типу зображення (зображення обличчя чи геометрія тіла) обирається пара зображень, за кожному з яких виконується розпізнавання. Після цього обчислюється відстань розпізнавання та робиться висновок щодо результату ідентифікації. Чим більш схожі між собою зображення однієї і тієї ж самої людини тим менше буде ця відстань.

4.3 Діаграма класів серверної частини системи

Для визначення складових серверної частини та їх взаємозв’язків створено діаграму програмних класів (рис. 4.5).

Клас ProcessImageDetails – відповідає за управління обробкою зображень.

Атрибути та методи класа ProcessImageDetails:

width: int – ширина зображення, у пікселах;

height: int – висота зображення, у пікселах;

framesToVerification: int – встановлено та верифіковано межу зображення, у разі перевищення повертається різниця;

imgLimit: int – лімітована межа зображення;

load(): void – метод для завантаження зображення;

update(): void – метод для оновлення зображення (викликається після обробки зображення);

checkConnection(): int – метод перевірки з’єднання;

reconnect(): void – метод для оновлення з’єднання.

Клас DetectionUtils містить налаштування та утіліти для ідентифікації зображення.

Атрибути та методи класа DetectionUtils:

destination: double – результат точності розпізнавання;

std: double – стандартне відхилення;

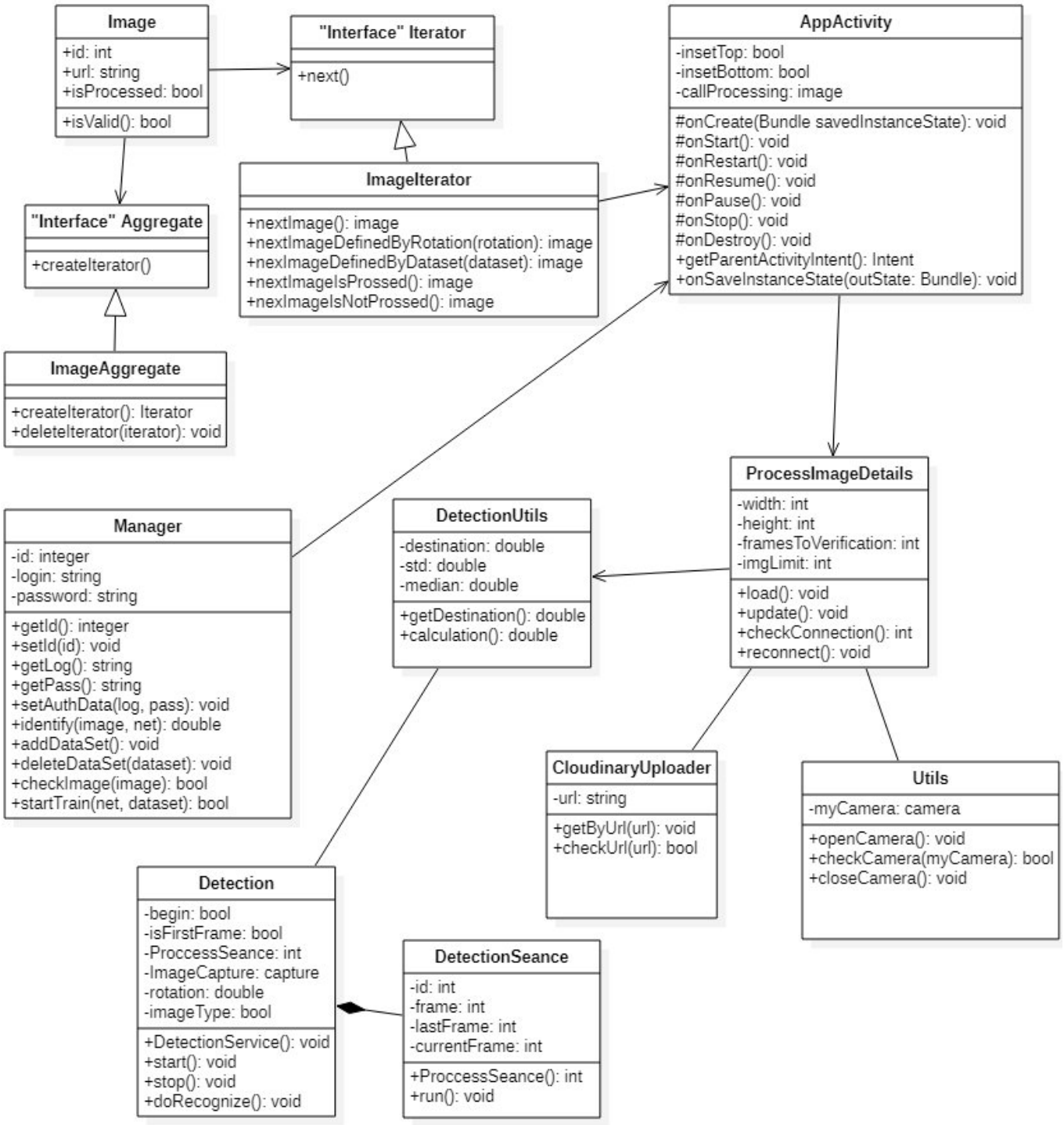

Рисунок 4.5 – Діаграма класів серверної частини системи

median: double – величина медіани;

getDestination(): double – метод для отримання тоності;

calculation(): double – метод для обчислення відстані розпізнавання.

Клас CloudinaryUploader використовується для роботи сервера з хмарним середовищем.

Атрибути та методи класа CloudinaryUploader:

url: string – посилання/адреса ресурсу;

getByUrl(url): void – метод для отримання зображення за url;

checkUrl(url): bool – метод для перевірки існування url.

Клас Utils – містить утіліти для роботи з камерою Андроід-пристрою.

Атрибути та методи класа Utils:

myCamera: camera – поточна використовувана камера пристрою;

openCamera(): void – метод для відкриття/підготовки до використання камери пристрою;

checkCamera(myCamera): bool – метод для перевірки доступу до камери;

closeCamera(): void – метод для закриття камери.

Клас Detection використовується для ідентифікації зображення.

Атрибути та методи класа Detection:

begin: bool – успішність початку ідентифікації;

isFirstFrame: bool – чи виконується перевірка з первинним зображенням (true) чи з результатом його обробки (false);

ProccessSeance: int – номер сеансу обробки;

ImageCapture: capture – капча (код) зображення;

rotation: double – кут відхилення від нормалі (при нормалі – людина розташована фронтально до камери);

imageType: bool – тип зображення (зображення обличчя – true, зображення тіла людини цілком - false);

DetectionService(): void – метод для підготовки до ідентифікації;

start(): void – метод початку процесу ідентифікації;

stop(): void – метод для зупинки процесу ідентифікації;

doRecognize(): void – метод для визначення значень зображення відповідно до базових метрик.

Клас DetectionSeance є складовою класа Detection. Клас DetectionSeance пов’язаний з класом Detection відношенням композиції.

Атрибути та методи класа DetectionSeance:

id: int – ідентифікатор сеансу;

frame: int – перший фрейм, що був використаний для ідентифікації;

lastFrame: int – останній фрейм, що був використаний для ідентифікації;

currentFrame: int – поточний фрейм;

ProccessSeance(): int – метод для визначення сеансу процеса ідентифікації;

run(): void – метод для запуску процеса.

Клас Manager використовується для роботи менеджера.

Атрибути та методи класа Manager:

id: integer – ідентифікатор менеджера;

login: string – логін менеджера;

password: string – пароль менеджера;

getId(): integer – метод для отримання ідентифікатору;

setId(id): void – метод для встановлення ідентифікатору;

getLog(): string – метод для отримання логіну менеджера;

getPass(): string – метод для отримання паролю менеджера;

setAuthData(log, pass): void – метод для встановлення аутентифікаційних даних менеджера;

identify(image, net): double – метод для ідентифікації зображення image з використанням нейромережі net;

addDataSet(): void – метод для додавання датасету;

deleteDataSet(dataset): void – метод для видалення датасету;

checkImage(image): bool – метод для перевірки зображення;

startTrain(net, dataset): bool – метод для початку тренування нейромережі.

4.4 Висновки до розділу

У четвертому розділі було виконано проектування серверної частини системи машинного зору. Розглянута загальна схема клієнт-серверної архітектури, проаналізовані переваги та недолікі такої архітектури.

Розроблена діаграма концептуальних класів системи машинного зору, наведений короткий опис класів.

Визначені ключові процеси системи, а саме тренування нейронної мережі з обраною топологією та виконання біометричної ідентифікації за зображеннями обличчя чи геометрії тіла людини. Для кожного процесу створено діаграми діяльності.

Для серверної частини системи створено діаграму класів, описано атрибути та методи класів, а також їх взаємозв’язки.

5 ПРОГРАМНА РЕАЛІЗАЦІЯ СЕРВЕРНОЇ ЧАСТИНИ СИСТЕМИ

5.1 Особливості використання інструментів програмування

У даній роботі демонструється серверна частина, взаємодія з клієнтами і спосіб організації даних, частина біометричної ідентифікації, яка працює з використанням нейромережі. Популярною і гнучкою платформою розробки серверних застосувань є мова програмування Java, розроблена компанією Oracle. Тому було прийнято рішення використовувати саме цю мову програмування для розробки серверної частини системи.

«У мові Java втілюється принцип об'єктно-орієнтованого програмування, оскільки Java в основному використовується для створення серверних програм і мобільного ПЗ. Також – це основа прикладних програм для системи Android.

Індекс TIOBE, який оцінює популярність світових мов програмування, на основі підрахунку результатів пошукових запитів, підніс Java на п'єдестал у 2018 році та віддав їй абсолютну першість серед програмістів країн СНД та ЄС.

Некомерційна організація Cloud Foundry Foundation (CFF) опублікувала рейтинг найбільш затребуваних мов програмування для корпоративних хмарних розробок. Згідно з їхнім дослідженням, лідером стала Java.

Завдяки віртуальній машині Java (JVM), написаний код можна запускати під Windows, Linux і MacOS. Це дозволяє реалізувати принцип «написана для одного – працює скрізь». Директор з розробки ПЗ компанії Twitter Роберт Бенсон називає адаптивність Java однією з головних причин, чому Twitter перейшла на JVM.

Крім названих причин, існують ще десятки додаткових переваг Java, через які з кожним роком все більше компаній переходять на цю передову мову програмування. Нею, наприклад, пишуть програми для Інтернету речей: програми для роботи з різного роду датчиками, камерами відеоспостереження тощо. Java популярна і в робототехніці.» [16]

Kotlin забезпечує кросплатформенний рівень для рідних программ, а також спільний доступ до логіки програм між веб-платформами, мобільними та настільними платформами, зберігаючи власний досвід для користувачів.

У якості серверної бази даних використано PostgresSQL.

«PostgreSQL - це система управління об'єктно-реляційними базами даних з відкритим кодом, ORDBMS, яка не належить або контролюється однією компанією чи особою. Оскільки програмне забезпечення postgresSQL є відкритим кодом, ним керують здебільшого за допомогою скоординованих онлайн-спроб активним світовим співтовариством розробників, ентузіастів та інших добровольців.

PostgreSQL підтримує майже всі функції реляційних баз даних і пропонує кілька незвичайних функцій, які зазвичай відсутні в інших двигунах RDBMS. Поширені об'єкти, що підтримуються, включають представлення даних, збережені процедури, індекси, тригери та визначені об'єктом типи даних, на додаток до загальних функцій RDBMS, таких як первинні ключі, зв'язки із зовнішніми ключами та атомність.

На відміну від інших баз даних, PostgreSQL не може бути створена без користувача. Для цього варто притримуватися такої послідовності:

1. Створити користувача PostgreSQL.
2. Створити БД.

Деякі критичні функції Postgresql схожі на Oracle DB та інші двигуни бази даних; такі функції включають використання таких понять, як табличні простори, точки збереження та відновлення в даний час.» [17]

5.2 Принципи роботи з камерою Андроід-пристрою

Для роботи з камерою повинні бути підключені відповідні бібліотеки:

```
import android.view.SurfaceHolder;
import android.view.SurfaceView;

import android.hardware.Camera;
import android.hardware.Camera.Size;
```

Спочатку потрібна ініцалізація об’єкта Camera:

```
camera = Camera.open();
```

Для створення превью використовується метод setPreviewDisplay:

```
SurfaceHolder surfaceHolder;
surfaceHolder = preview.getHolder();
camera.setPreviewDisplay(surfaceHolder);
```

Для роботи з Surface (створення, зміна та видалення) реалізовано наступні методи:

```
public void surfaceCreated(SurfaceHolder holder);
public void surfaceChanged(SurfaceHolder holder, int format, int width, int height);
public void surfaceDestroyed(SurfaceHolder holder);
```

Налаштування ширини та висоти зображення:

```
LayoutParams lp = preview.getLayoutParams();
lp.width = ширина;
lp.height = висота;
preview.setLayoutParams(lp);
```

Ресурси - один з основних компонентів, з якими потрібно працювати дуже часто. В Android прийнято тримати зображення об'єкти поза вихідного коду. Система підтримує зберігання ресурсів окремих файлах. Ресурси легко підтримувати, оновлювати, редагувати.

Налаштування файлу main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout android:id="@+id/FrameLayout01"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <SurfaceView android:id="@+id/SurfaceView01"
        android:layout_width="wrap_content" android:layout_height="wrap_content">
    </SurfaceView>
    <Button android:text="@+id/Button01" android:id="@+id/Button01"
        android:layout_width="wrap_content" android:layout_height="wrap_content">
    </Button>
     </FrameLayout>
```

Налаштування файлу AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="test.camera"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name"
android:debuggable="true">
        <activity android:name=".MainScreen" android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   </manifest>
```

5.3 Висновки до розділу

У розділі 5 розглянуто особливості розробки програмних застосувань з використанням мови програмування Java. Наведено принципи роботи з камерою Андроід-пристрою: ініціалізація, налаштування, створення зображення та його видалення.

6 ТЕСТУВАННЯ РОБОТИ СИСТЕМИ

6.1 Сценарії тестування

Для того, щоб перевірити виконання функціональних вимог до системи машинного зору, потрібно виконати функціональне тестування [18]. Для цього спочатку потрібно створити тестові сценарії (табл. 6.1), а потім визначити конкретні результати для кожного з них.

Таблиця 6.1 – Тестові сценарії системи машинного зору

№	Тестовий сценарій	Очікувана реакція	Мета проведення тесту
ТС1	Реєстрація	Створення профілю для нового менеджера	Перевірити можливість менеджера зареєструватись у системі
ТС2	Авторизація	Авторизація профілю менеджера	Перевірити, що менеджер може працювати з системою
ТС3	Завантаження датасетів зображень	Менеджер завантажив нові датасети	Переконатися в тому, що менеджер може розширювати набір датасетів.
ТС4	Перевірка датасету зображень	Перевірка придатності ата сету для тренування нейромережі	Перевірити, що обраний менеджером датасет може бути використаний для тренування

Продовження таблиці 6.1

№	Тестовий сценарій	Очікувана реакція	Мета проведення тесту
ТС5	Тренування нейронної мережі	Мережа натренована на зображеннях датасету	Перевірити, що менеджер може задавати зображення для тренування мережі.
ТС6	Оцінка якості біометричної ідентифікації	Менеджер отримує оцінку результату ідентифікації	Переконатися в тому, що нейромережі вміє оцінювати якість ідентифікації за порівнянням двох зображень.
ТС7	Завантажити зображення	Користувач завантажує зображення для ідентифікації	Перевірити можливість завантаження зображень та перевірки їх формату
ТС8	Створити зображення	Користувач створює зображення для ідентифікації	Перевірити можливість створення зображень з доступом до камери пристрою
ТС9	Виконати попередню обробку зображення	Користувач покращує якість зображення	Перевірити кінцеву можливість підвищення точності розпізнавання

Продовження таблиці 6.1

№	Тестовий сценарій	Очікувана реакція	Мета проведення тесту
ТС10	Отримати результат біометричної ідентифікації	Користувач отримує підсумковий результат ідентифікації	Переконатися в тому, що нейромережі вміє використовувати декілька зображень для підвищення точності результату.
ТС11	Збереження результату	Користувач зберігає результат ідентифікації	Перевірити можливість збереження даних у хмарному середовищі

6.2. Функціональне тестування системи машинного зору

Розглянемо функціональне тестування для декількох сценаріїв: «Тренування нейронної мережі», «Оцінка якості біометричної ідентифікації» та «Отримати результат біометричної ідентифікації» (табл. 6.2 – 6.4).

Таблиця 6.2 – Тестові випадки для сценарію «Тренування нейронної мережі».

№	Дія	Очікуваний результат	Результат тестування
1	Менеджер робить запит на тренування мережі	Система надає форму інтерфейсу	Успішно
2	Менеджер обирає датасет, за яким потрібно тренувати мережу	Система перевіряє дані датасету	Успішно

Продовження таблиці 6.2

№	Дія	Очікуваний результат	Результат тестування
3	Менеджер налаштовує топологію мережі	Система перевіряє дані топології	Успішно
4	Менеджер запускає процес тренування	Система виконує тренування та повертає результат	Успішно
5	Менеджер переглядає результат тренування. Інтернет-зв’язок стабільний.	Результати відображені у формі інтерфейсу	Успішно
6	Менеджер переглядає результат тренування. Інтернет-зв’язок нестабільний.	Повідомлення про проблеми з'єднання	Успішно

Таблиця 6.3 – Тестові випадки для сценарію «Оцінка якості біометричної ідентифікації».

№	Дія	Очікуваний результат	Результат тестування
1	Система запускає процес тестування нейромережі	Повідомлення про успішний запуск	Успішно
2	Система перевіряє цілісність обраних зображень	Повідомлення у разі помилки	Успішно
3	Система розраховує оцінку точності	Повідомлення про оцінку	Успішно
4	Зображення, призначені для тестування, недоступні	Повідомлення про помилку	Успішно

Продовження таблиці 6.3

№	Дія	Очікуваний результат	Результат тестування
5	Менеджер переглядає результат оцінювання. Інтернет-зв’язок нестабільний.	Повідомлення про проблеми з'єднання	Успішно

Таблиця 6.4 – Тестові випадки для сценарію «Отримати результат біометричної ідентифікації».

№	Дія	Очікуваний результат	Результат тестування
1	Система перевіряє чи є зображення для проведення ідентифікації	Повідомлення у разі помилки	Успішно
2	Система запускає процес ідентифікації	Повідомлення про успішний запуск	Успішно
3	Інтернет-зв’язок стабільний, менеджер переглядає результат	Повідомлення про точність розпізнавання	Успішно
4	Інтернет-зв’язок нестабільний, менеджер переглядає результат	Повідомлення про проблеми з'єднання	Успішно

6.3 Тестування топології нейронної мережі

Для того, щоб обрати топологію нейронної мережі, яка дозволяє найбільш точно ідентифікувати зображення, проведемо серію експериментів. Для датасету, який містить тільки фронтальні зображення, будемо змінювати розміри згорткових шарів (ЗШ), підвибіркових шарів (ПШ), а також ядер. Для кожного випадку топології будемо оцінювати точність розпізнавання зображень.

Спочатку розглянемо датасет з зображеннями обличчя людини. Наведемо декілька випадків у порядку зростання загальної точності ідентифікації:

1) ЗШ1 24*24, ЗШ2 24*24, ПШ1 24*24, ПШ1 20*20, Я1 10*10, Я2 5*5, Я3 10*10, Я4 5*5.

Як можна побачити на рис. 6.1, точність є вкрай низькою та коливається у діапазоні від 36,3% до 64,8%. Максимальний розмах складає 29,4%.

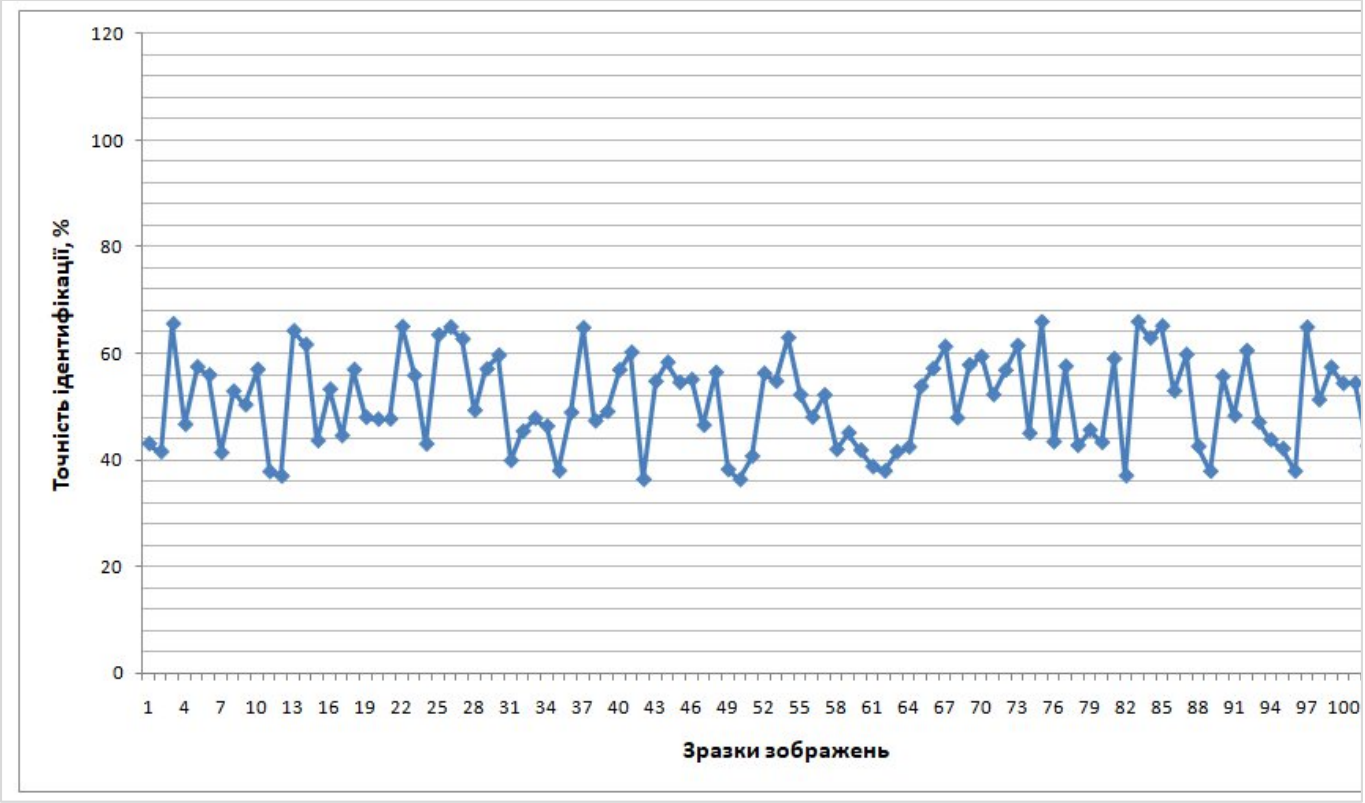

Рисунок 6.1 – Результати експерименту для топології 1

2) ЗШ1 24*24, ЗШ2 10*10, ПШ1 24*24, ПШ1 20*20, Я1 10*10, Я2 5*5, Я3 10*10, Я4 5*5.

На рис. 6.2 можна побачити, що для ряда випадків точність зросла, але збільшився розмах. Тепер точність коливається у діапазоні від 45,1% до 79,9%. Максимальний розмах складає 34,6%.

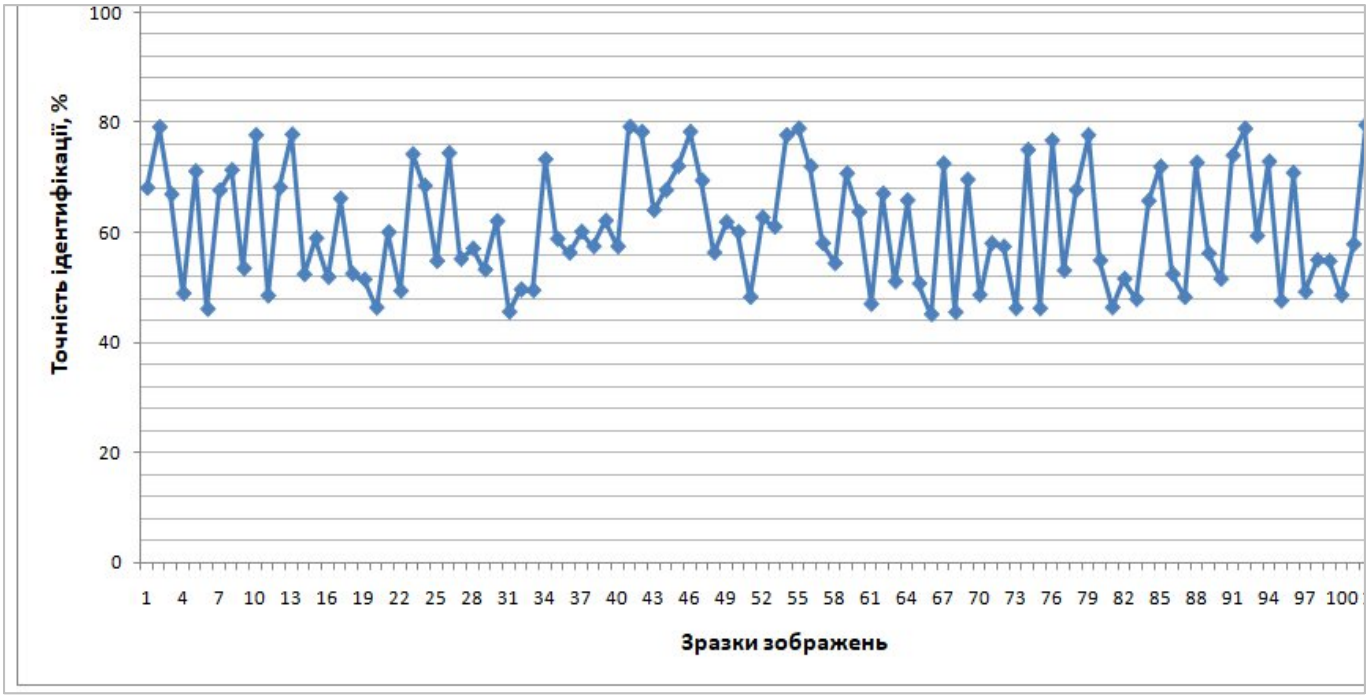

Рисунок 6.2 – Результати експерименту для топології 2

3) ЗШ1 24*24, ЗШ2 10*10, ПШ1 12*12, ПШ1 10*10, Я1 10*10, Я2 5*5, Я3 10*10, Я4 5*5.

У цьому варіанті топології зменшено розмір карт на підвибіркових шарах, інші параметри без змін.

На рис. 6.3 видно, що у цілому точність зросла та зменшився максимальний розмах. Тепер точність коливається у діапазоні від 65,3% до 86,0%. Максимальний розмах складає 20,7%.

4) ЗШ1 24*24, ЗШ2 10*10, ПШ1 12*12, ПШ1 5*5, Я1 4*4, Я2 2*2, Я3 4*4, Я4 2*2.

При даній топології зменшено розмір ядер, інші параметри не змінились.

На рис. 6.4 видно, що у цілому точність зросла, але збільшився розмах. Тепер точність коливається у діапазоні від 79,0% до 89,9%. Максимальний розмах складає 10,8%.

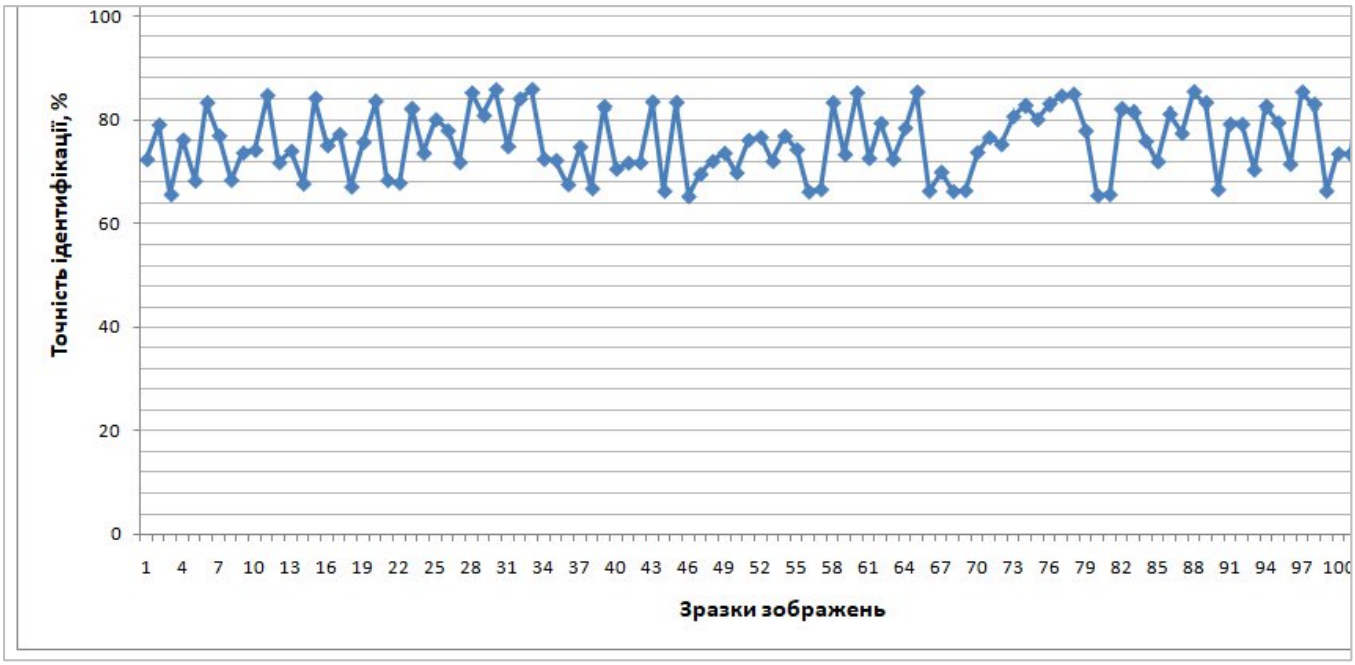

Рисунок 6.3 – Результати експерименту для топології 3

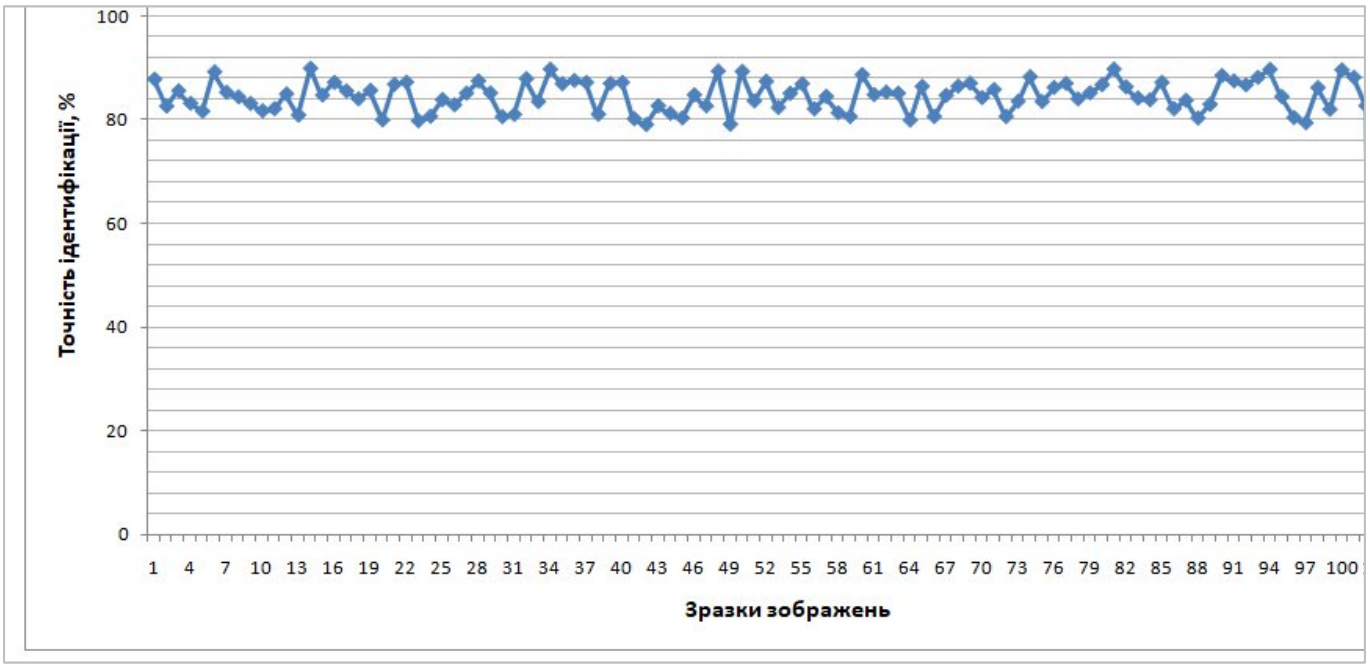

Рисунок 6.4 – Результати експерименту для топології 4

Отримані результати можна вважати досить високими, тому приймемо саме цю топологію згорткової нейронної мережі для ідентифікації людини за зображенням обличчя.

Тепер розглянемо як обрана нейромережі може ідентифікувати людину за геометрією тіла.

На рис. 6.5 наведений відповідний графік точності. Як можна побачити, вона є трохи меншою, ніж у попередньому випадку (для ідентифікації за обличчям) – точність коливається у діапазоні від 72,1% до 84,8%, максимальний розмах складає 12,7%.

Однак це взагалі є типовою ситуацією, тобто незалежно від обраного алгоритму ідентифікації за інших рівних умов ідентифікація за обличчям є більш точною, ніж за геометрією тіла (соматотипом).

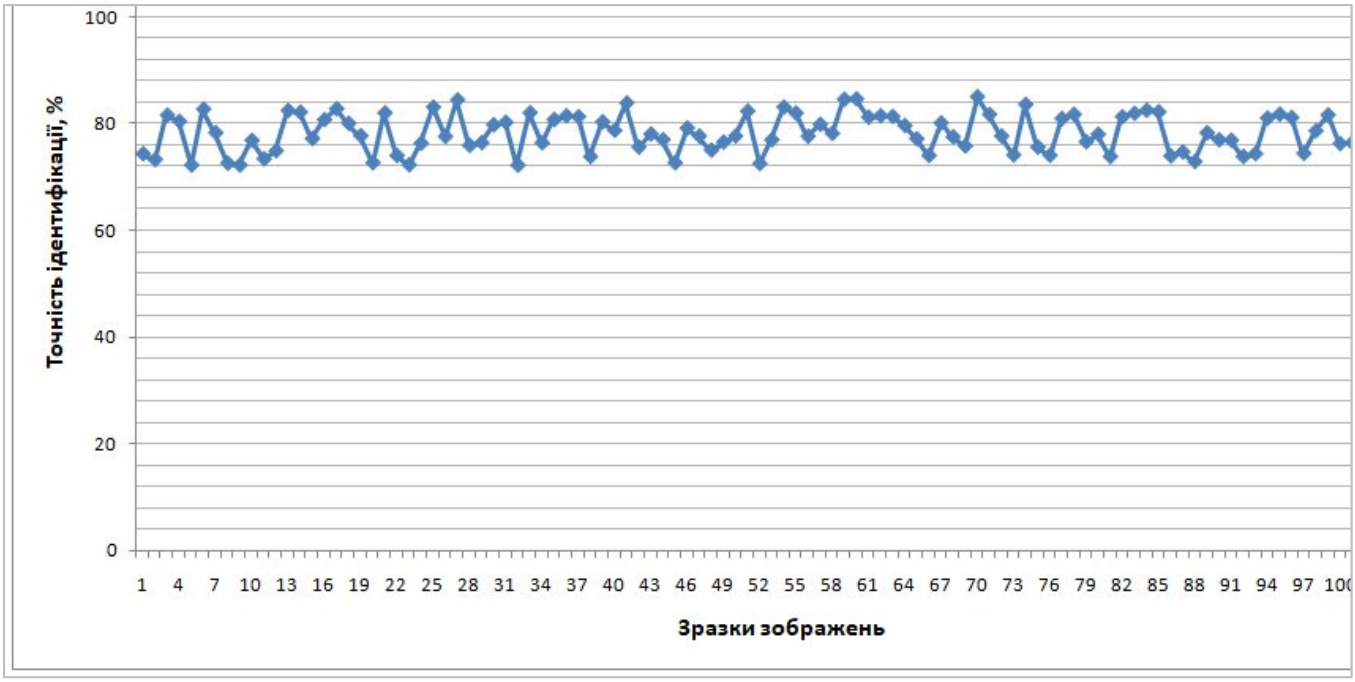

Рисунок 6.5 – Результати експерименту для топології 4

Для ідентифікації людини за геометрією тіла також був проведений ряд експериментів. Використовувались нейронні мережі з топологами, які відповідають експериментам № 1, 2 та 3. Однак отримані результати показали, що остання топологія (експерименти № 4 та 5) є найкращою та забезпечує найвищу точність ідентифікації людини.

6.3 Висновки до розділу

У розділі 6 проведено тестування роботи розробленої системи. Для цього спочатку обрані тестові сценарії, які відповідають варіантам використання. Для кожного тестового сценарію вказана очікувана реакція та мета проведення тесту.

Після цього виконано функціональне тестування за всіма сценаріями. Наведені тестові випадки для сценаріїв «Тренування нейронної мережі», «Оцінка якості біометричної ідентифікації» та «Отримати результат біометричної ідентифікації».

Виконано тестування топології нейронної мережі на необроблених фронтальних зображеннях. За результатами експерименту обрана топологія, що надає максимальну точність 89,9% для ідентифікації за обличчям людини та 84,8% для ідентифікації за геометрією тіла. Для подальшого підвищення точності потрібна попередня обробка зображень.

ВИСНОВКИ

У даній кваліфікаційній роботі розроблено серверну частину програмної системи машинного зору для біометричної ідентифікації осіб за двовимірними зображеннями з використанням методів штучного інтелекту. Використання системи дозволило забезпечити точність ідентифікації за обличчям людини 89,9% та за геометрією тіла 84,8% на фронтальних попередньо необроблених зображеннях.

У першому розділі розглянуто сучасний стан проблеми біометричної ідентифікації, розглянуто архітектурні ідеї згорткових мереж. Визначено базові метрики біометричної ідентифікації, що базуються на зображенні обличчя людини та на основі даних геометрії тіла.

У другому розділі визначена топологія узагальненої нейронної мережі для біометричної ідентифікації людини за зображеннями обличчя та за соматотипом. Створений екземпляр згорткової нейронної мережі для RGB-зображень.

У третьому розділі створено функціональні та нефункціональні вимоги до серверної частини розроблюваної системи.

Четвертий розділ містить проектування серверної частини системи машинного зору та базі клієнт-серверної архітектури.

У п’ятому розділі розглянуто особливості розробки програмних застосувань та наведено принципи роботи з камерою Андроід-пристрою.

Шостий розділ містить тестування роботи розробленої системи з використанням тестових сценаріїв та тестових випадків для відповідних сценаріїв. Виконано тестування топології нейронної мережі.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Біометрична ідентифікація [Електроний ресурс] https://sites.google.com – Режим доступу: https://sites.google.com/site/identifikaciataautentifikacia/ponatta-pro-identifikaciu/biometricna-identifikacia

2. Біометрична ідентифікація [Електроний ресурс] https://infocom.ua/ Режим доступу: https://infocom.ua/services-for-home/biometric-identification/

3. Згорткова нейронна мережа – просте пояснення CNN та її застосування [Електроний ресурс] https://evergreens.com.ua – Режим доступу: https://evergreens.com.ua/ua/articles/cnn.html

4. Sean Clarkson, Jon Wheat, Ben Heller & Simon Choppin. Assessing the Suitability of the Microsoft Kinect for Calculating Person Specific Body Segment Parameters. 2015. Computer Vision - ECCV Workshops pp 372–385.

5. R-CNN Quick Overview [Електроний ресурс] https://blog.paperspace.com/ – Режим доступу: https://blog.paperspace.com/faster-r-cnn-explained-object-detection/

6. S. A. Toki, S. Rahman, S. M. Billah Fahim, A. Al Mostakim and M. K. Rhaman, "RetinalNet-500: A newly developed CNN Model for Eye Disease Detection," 2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), 2022, pp. 459-463.

7. Jifeng Dai, Yi Li, Kaiming He, Jian Sun. Region-based Fully Convolutional Network. Computer Vision and Pattern Recognition. 2016.

8. YOLO — You only look once, real time object detection explained [Електроний ресурс] https://towardsdatascience.com/ – Режим доступу: https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006

9. Вивчення причинно-наслідкових взаємозв’язків методом Ісікави [Електроний ресурс] http://market.avianua.com/ – Режим доступу: http://market.avianua.com/?p=875

10. «Дерево цілей» підприємства [Електроний ресурс] https://buklib.net/ – Режим доступу: https://buklib.net/books/25617/

11. Що таке дерево цілей [Електроний ресурс] https://pyrogiv.kiev.ua/ – Режим доступу: https://pyrogiv.kiev.ua/shho-take-derevo-cilej/

12. Формати графічних файлів : основні типи [Електроний ресурс] https://smartik.kiev.ua/ – Режим доступу: https://smartik.kiev.ua/formaty-hrafichnykh-fajliv-osnovni-typy/

13. Формати графічних файлів [Електроний ресурс] http://www.ni.biz.ua/ – Режим доступу: http://www.ni.biz.ua/3/3_6/3_65306_formati-graficheskih-faylov.html

14. Клієнт-серверна архітектура та ролі серверів [Електроний ресурс] https://medium.com/ Режим доступу: https://medium.com/@IvanZmerzlyi/

15. Архітектура клієнт-сервер [Електроний ресурс] http://inter.ptngu.com/ – Режим доступу: http://inter.ptngu.com/kompyuterni-merezhi/arhitektura-kliyent-server

16. Чому Java – найпопулярніша мова програмування у світі [Електроний ресурс] https://java.lviv.ua/chomu-java-najpopulyarnisha-mova-programuvannya-u-sviti

17. Що таке Postgresql? [Електроний ресурс] https://uk.theastrologypage.com/postgresql

18. Software Engineering – Product Quality – Part 4: Quality in use metric [Електроний ресурс] /ISO/IECTR 9126-4:2004. Режим доступу: www / URL:

https://www.iso.org/obp/ui/#iso:std:iso-iec:tr:9126:-4:ed-1:v1:en

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
package com.identification.bezruchenko.identification;

import com.fasterxml.jackson.annotation.JsonInclude;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Verification {

private Long verificationId;

    private long imageId;

    private String clientId;

    private VerificationType type;

    private long timestamp = -1;

@JsonInclude(JsonInclude.Include.NON_NULL)
private VerificationDetails details;

    public Verification withDetails(VerificationDetails details) {
this.details = details;
        return this;
}

public Verification withImageId(long camId) {
this.imageId = camId;
        return this;
}

public Verification withClientId(String str) {
this.clientId = str;
        return this;
}
}
================================================================================
package com.identification.bezruchenko.verification;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class VerificationDetails {

@JsonIgnore
private Long detailId;
```

```
@JsonIgnore
private Long verificationId;

    private String pictureUrl;

    public VerificationDetails withVerificationId(long verificationId) {
this.verificationId = verificationId;
        return this;
}

private String videoUrl;
}
================================================================================
package com.identification.bezruchenko.verification;


import com.fasterxml.jackson.annotation.JsonValue;

public enum ImageType {
FACE, BODY;

@JsonValue
public String getValue() {
return this.name().toLowerCase();
}
}
================================================================================
package com.identification.bezruchenko.verification;

import lombok.RequiredArgsConstructor;
import org.springframework.messaging.rsocket.RSocketRequester;
import org.springframework.stereotype.Component;
import reactor.core.publisher.Mono;

@Component
@RequiredArgsConstructor
public class RSocketVerificationService {

private final RSocketRequester metadataRequester;

    public Mono<Boolean> createVerification(Verification verification) {
return metadataRequester.route("verification.create")
                .data(verification)
                .retrieveMono(Boolean.class);
}
}
================================================================================
package com.identification.bezruchenko.blur;

import org.opencv.core.Mat;

import java.util.concurrent.atomic.AtomicInteger;

public class BurDetectionService {
private final AtomicInteger blurredFrames = new AtomicInteger(0);

    private int frame = 0;
```

```
    public boolean blurDetectionVerification(Mat currentFrame, double blurLimit, int
framesToVerification) {
if(burredFrames.get()!=0 || frame==0) {
if (burLimit > BurDetectionUtils.getLaplacianIndex(currentFrame)) {
if (burredFrames.get() == framesToVerification) {
burredFrames.set(0);
                    return true;
} else burredFrames.incrementAndGet();
} else burredFrames.set(0);
frame = 4;
} else{
frame--;
}
return false;
}



}
package com.identification.bezruchenko.blur;

import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfDouble;
import org.opencv.imgproc.Imgproc;

public class BurDetectionUtils {
public static double getLaplacianIndex(Mat in){

        Mat matGray=new Mat();
Mat destination = new Mat();
MatOfDouble std= new MatOfDouble();
MatOfDouble median = new MatOfDouble();

Imgproc.cvtColor(in, matGray, Imgproc.COLOR_BGR2GRAY);
Imgproc.Laplacian(matGray, destination, 3);


Core.meanStdDev(destination, median , std);

        return Math.pow(std.get(0,0)[0],2);
}
}=============================================================================

package com.identification.bezruchenko.bur;

import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfDouble;
import org.opencv.imgproc.Imgproc;

public class BlurDetectionUtils {
public static double getLaplacianIndex(Mat in){

        Mat matGray=new Mat();
Mat destination = new Mat();
MatOfDouble std= new MatOfDouble();
MatOfDouble median = new MatOfDouble();
```

```
Imgproc.cvtColor(in, matGray, Imgproc.COLOR_BGR2GRAY);

Core.meanStdDev(destination, median , std);

        return Math.pow(std.get(0,0)[0],2);
}
}
```

```
package com.identification.bezruchenko.image;

import com.fasterxml.jackson.annotation.JsonInclude;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.With;

@Data
@NoArgsConstructor
@AllArgsConstructor
@With
public class Image {

private long imageId;

    private String url;

    private String name;

    private String clientId;


@JsonInclude(JsonInclude.Include.NON_NULL)
private ImageDetails details;
}
```

```
package com.identification.bezruchenko.image;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class ImageDetails {
@JsonIgnore
private long imageId;

@JsonIgnore
private long detailId;


    private Integer burSkip;

    private Integer framesToVerificationBur;

    private Integer framesToVerification;

    private Double psnrLimit;
```

```
    private Double burLimit;

    private Double fps;

    private Integer width;

    private Integer height;



    public ImageDetails withImageId(long id) {
this.imageId = id;
        return this;
}

public ProcessImageDetails(long imageId, //constructor without id
Double fps,
Integer width,
Integer height,
Integer burSkip,
Integer framesToVerificationBur,
Integer framesToVerificationMotion,
Double psnrLimit,
Double burLimit) {
this.imageId = imageId;
        this.fps = fps;
        this.width = width;
        this.height = height;
        this.burSkip = burSkip;
        this.framesToVerificationBur = framesToVerificationBur;
        this.framesToVerificationMotion = framesToVerificationMotion;
        this.psnrLimit = psnrLimit;
        this.burLimit = burLimit;
}
}================================================================================
package com.identification.bezruchenko.image;

import lombok.RequiredArgsConstructor;
import org.springframework.messaging.rsocket.RSocketRequester;
import org.springframework.stereotype.Component;
import reactor.core.publisher.Mono;

@Component
@RequiredArgsConstructor
public class RSocketService {

private final RSocketRequester metadataRequester;

    public Mono<Image> getImage(long camId) {
return metadataRequester.route("image.getById")
                .data(camId)
                .retrieveMono(Image.class);
}
}================================================================================
package com.identification.bezruchenko.detection;

import com.cloudinary.utils.ObjectUtils;
```

```
import java.io.File;
import java.io.IOException;
import java.util.Map;

public class CloudinaryUploader {
public static String getURL(String filename){
        File toUpload = new File(filename);
        return upload(toUpload);
}
private static String upload(File file){
        String urlImage = "";
        try {
            Map                                uploadResult                                =
CloudinaryConst.getCloudinary().uploader().upload(file, ObjectUtils.emptyMap());
urlImage = uploadResult.get("url").toString();
} catch (IOException e) {
            e.printStackTrace();
}
return urlImage;
}
boolean checkConnection(){
return true;
}
}=========================================================================
package com.identification.bezruchenko.detection;

import com.identification.bezruchenko.verification.Verification;
import com.identification.bezruchenko.verification.VerificationDetails;
import com.identification.bezruchenko.verification.VerificationType;
import com.identification.bezruchenko.verification.RSocketVerificationService;
import com.identification.bezruchenko.bur.BurDetectionService;
import com.identification.bezruchenko.image.Image;
import com.identification.bezruchenko.motion.MotionDetectionUtils;
import com.identification.bezruchenko.motion.MotionDetectionHandler;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.SneakyThrows;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import java.io.File;
import java.io.IOException;

@NoArgsConstructor
@Component
@Scope("prototype")
public class DetectionFace {
@Autowired
private RSocketVerificationService verificationService;

    private Boolean begin = false;
    private Boolean isFirstFrame = true;
    private ProccessSeance Seance = null;
    private ImageCapture Image = new ImageCapture();
```

```
    private     final     BurDetectionService     burDetectionService     =     new
BurDetectionService();
    private final MDetectionHandler MDetectionHandler = new MDetectionHandler();

    public void start(Image image, DetectionParameters parameters) throws Exception {
if (!begin) {
try{
if(!image.getUrl().equals("0")) {
Image = new ImageCapture(image.getUrl(), Imageio.CAP_FFMPEG);
} else {
Image = new ImageCapture(Integer.parseInt(image.getUrl()));
}

            } catch (Exception e){
                e.printStackTrace();
}

if (Image.isOpened()) {
isFirstFrame = true;
Seance = new ProccessSeance(image, parameters);
Seance.start();
begin = true;
} else {
throw new Exception("Unsuccessful start for image " + image.getUrl());
}
        }
    }

public void update(DetectionParameters parameters) {
Seance.setParameters(parameters);
}

public void stop() {
if (begin) {
try {
begin = false;
Seance.interrupt();
} catch (Exception ex) {
                ex.printStackTrace();
}
Image.release();

}
    }

@Setter
class ProccessSeance extends Seance {
        Image image;
DetectionParameters parameters;
        private final Mat frame;
        private final Mat lastFrame;
        private final Mat currentFrame;
        private final Mat frameRaw = new Mat();
        public ProccessSeance(Image image, DetectionParameters detectionParameters) {
this.image = image;
            this.parameters = detectionParameters;
frame   =   new   Mat(image.getDetails().getHeight(),   image.getDetails().getWidth(),
CvType.CV_8UC3);
```

```
lastFrame = new Mat(image.getDetails().getHeight(), image.getDetails().getWidth(),
CvType.CV_8UC3);
currentFrame = new Mat(image.getDetails().getHeight(), image.getDetails().getWidth(),
CvType.CV_8UC3);
}

@SneakyThrows
        @Override
public void run() {
if (dataset.isOpened()) {
while (begin) {
do {
if(!dataset.read(frameRaw)){
continue;
}
if (frameRaw.size().height >0 &&frameRaw.size().width >0) {
                            Imgproc.resize(frameRaw, frame, frame.size());
}
                    } while (!isFirstFrame && MDetectionUtils.isChanged(frame,
lastFrame) && !Seance.interrupted());

frame.copyTo(currentFrame);

                    if (isFirstFrame) {
frame.copyTo(lastFrame);
isFirstFrame = false;
                        continue;
}
if (parameters.isCheckBiometrical()) {
if(MDetectionHandler.moveDetectionVerification(currentFrame, lastFrame,
image.getDetails().getPsnrLimit(), image.getDetails().getFramesToVerificationM())) {
                            createVerification(VerificationType.M_DETECTED);
}
                    }
if (parameters.isCheckBur()) {
if (burDetectionService.burDetectionVerification(frame,
image.getDetails().getBurLimit(), image.getDetails().getFramesToVerificationBur())) {
                            createVerification(VerificationType.BURRED);
}
                    }
frame.copyTo(lastFrame);
}
            }
        }

private void createVerification(VerificationType type) throws IOException {
            File file = File.createTempFile("Detection", ".jpg");
Imgcodecs.imwrite(file.getAbsolutePath(), frame);
verificationService.createVerification(new Verification(null, image.getImageId(),
image.getClientId(), type,
System.currentTimeMillis(),
                    new VerificationDetails(null, null,
CloudinaryUploader.getURL(file.getAbsolutePath())))
                    .subscribe();
file.delete();
}
    }
}==========================================================================
package com.identification.bezruchenko.detection;
```

```
import com.identification.bezruchenko.image.RSocketService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.ObjectProvider;
import org.springframework.stereotype.Component;
import reactor.core.publisher.Mono;

import java.util.HashMap;
import java.util.Map;

@Component
@RequiredArgsConstructor
public class DetectionService {

private final RSocketService imageService;

    private final Map<Long, DetectionStatus>runs = new HashMap<>();

    private final ObjectProvider<DetectionFace>provider;

    public Mono<Boolean>handle(DetectionParameters param) {
return imageService.getImage(param.getImageId())
                .flatMap(cam -> {
if (runs.containsKey(param.getImageId())) {
if (!param.isEnable()) {
runs.get(param.getImageId()).stop();
runs.remove(param.getImageId());
} else {
runs.get(param.getImageId()).update(param);
}
                    } else if (param.isEnable()) {
                DetectionStatus   status   =   new   DetectionStatus(cam,   param,
provider.getObject());
status.start();
runs.put(param.getImageId(), status);
}
return Mono.just(Boolean.TRUE);
});
}
}==========================================================================
package com.identification.bezruchenko.ident;

import com.identification.bezruchenko.image.RSocketService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.ObjectProvider;
import org.springframework.stereotype.Component;
import reactor.core.publisher.Mono;

import java.util.HashMap;
import java.util.Map;

@Component
@RequiredArgsConstructor
public class DetectionService {

private final RSocketService imageService;

    private final Map<Long, DetectionStatus>runs = new HashMap<>();
```

```
    private final ObjectProvider<DetectionFace>provider;

    public Mono<Boolean>handle(DetectionParameters param) {
return imageService.getImage(param.getImageId())
                .flatMap(cam -> {
if (runs.containsKey(param.getImageId())) {
if (!param.isEnable()) {
runs.get(param.getImageId()).stop();
runs.remove(param.getImageId());
} else {
runs.get(param.getImageId()).update(param);
}
                    } else if (param.isEnable()) {
                        DetectionStatus status = new DetectionStatus(cam, param,
provider.getObject());
status.start();
runs.put(param.getImageId(), status);
}
return Mono.just(Boolean.TRUE);
});
}
}
```

```
package com.identification.bezruchenko.utils;

import javafx.embed.swing.SwingFXUtils;
import javafx.scene.image.Image;
import javafx.scene.image.PixelReader;
import javafx.scene.image.WritablePixelFormat;
import org.opencv.core.CvType;
import org.opencv.core.Mat;

import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;
import java.nio.ByteBuffer;

public class Utils {
    public static Image matToImage(Mat frame) {
        try {
            return SwingFXUtils.toFXImage(matToBufferedImage(frame), null);
        } catch (Exception e) {
            System.err.println("Cannot convert the Mat object: " + e);
            return null;
        }
    }

    public static Mat imageToMat(Image image) {
        int width = (int) image.getWidth();
        int height = (int) image.getHeight();
        byte[] buffer = new byte[width * height * 4];

        PixelReader reader = image.getPixelReader();
        WritablePixelFormat<ByteBuffer> format =
WritablePixelFormat.getByteBgraInstance();
        reader.getPixels(0, 0, width, height, format, buffer, 0, width * 4);

        Mat mat = new Mat(height, width, CvType.CV_8UC4);
        mat.put(0, 0, buffer);
        return mat;
```

```
    }

    private static BufferedImage matToBufferedImage(Mat original) {
        int width = original.width(), height = original.height(), channels =
original.channels();
        byte[] sourcePixels = new byte[width * height * channels];
        original.get(0, 0, sourcePixels);

        BufferedImage image;
        if (original.channels() > 1) {
            image = new BufferedImage(width, height, BufferedImage.TYPE_3BYTE_BGR);
        } else {
            image = new BufferedImage(width, height, BufferedImage.TYPE_BYTE_GRAY);
        }
        final byte[] targetPixels = ((DataBufferByte)
image.getRaster().getDataBuffer()).getData();
        System.arraycopy(sourcePixels, 0, targetPixels, 0, sourcePixels.length);

        return image;
    }
}
```

```
package com.identification.bezruchenko.utils

import java.util.ArrayList;
import java.util.List;

import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.Point;
import org.opencv.core.RotatedRect;
import org.opencv.core.Scalar;
import org.opencv.imgproc.Imgproc;
import org.opencv.core.MatOfPoint;
import org.opencv.core.MatOfPoint2f;

public class Preprocessing {

      public Preprocessing() {}

      public Mat applyThreshold(Mat image, int threshValue)
      {
            Mat threshImage = new Mat();
            Imgproc.threshold(image, threshImage, threshValue, 255,
Imgproc.THRESH_BINARY);

            return threshImage;
      }

      public Mat applyMorphology(Mat image, int kernelSize, int iter, String type)
      {
            Mat morphImage = new Mat();
            Mat kernel = Mat.ones(kernelSize, kernelSize, CvType.CV_32F);
            if(type.equals("erode"))
                  Imgproc.erode(image, morphImage, kernel, new Point(-1,-1) ,iter);
            else
                  Imgproc.dilate(image, morphImage, kernel, new Point(-1,-1) ,iter);
```

```
		return morphImage;
	}

	public Mat convertHSVFormat(Mat frame,int h_low, int h_high, int s_low, int
s_high, int v_low, int v_high)
	{
		Mat image = new Mat();
		Mat result = new Mat();
		Imgproc.cvtColor(frame, image, Imgproc.COLOR_BGR2HSV);

		Scalar lower = new Scalar(h_low, s_low, v_low);
		Scalar upper = new Scalar(h_high, s_high, v_high);

		Core.inRange(image, lower, upper, result);

		return result;
	}

	public List<MatOfPoint> findContours(Mat threshImage)
	{
		Mat canny = new Mat();
		Imgproc.Canny(threshImage, canny, 100, 200);

		List<MatOfPoint> contours = new ArrayList<>();
		Mat hierarchy = new Mat();
		Imgproc.findContours(canny, contours, hierarchy, Imgproc.RETR_TREE,
Imgproc.CHAIN_APPROX_SIMPLE);

		return contours;
	}

	public MatOfPoint getMaxRectangle(List<MatOfPoint> contours)
	{
		double maxArea=0, currentArea=0;
		MatOfPoint maxContour = null;

		for(MatOfPoint c : contours)
		{
			currentArea = Imgproc.minAreaRect(new
MatOfPoint2f(c.toArray())).size.area();
			if(currentArea > maxArea)
			{
				maxArea = currentArea;
				maxContour = c;
			}
		}

		return maxContour;
	}

	public Mat drawRectangle(MatOfPoint rect, Mat image)
	{
		RotatedRect rectangle = Imgproc.minAreaRect(new
MatOfPoint2f(rect.toArray()));
		Point[] points = new Point[4];
		rectangle.points(points);
		for(int i=0; i<4; ++i)
			Imgproc.line(image, points[i], points[(i+1)%4], new
Scalar(255,255,255));
```

```
            return image;
      }


      private Point[] sortPointArray(Point[] points)
      {
            Point[] sorted = new Point[points.length];
            Numeric num = new Numeric();

            double[] summed = num.sumPoints(points);
            sorted[0] = points[num.argMin(summed)];
            sorted[2] = points[num.argMax(summed)];

            double[] diff = num.diffPoints(points);
            sorted[1] = points[num.argMax(diff)];
            sorted[3] = points[num.argMin(diff)];

            return sorted;
      }

      public double calculateAngle(MatOfPoint contour)
      {
            RotatedRect rectangle = Imgproc.minAreaRect(new
MatOfPoint2f(contour.toArray()));

            return rectangle.angle;
      }
      public double calculateDistance(MatOfPoint contour)
      {
            RotatedRect rectangle = Imgproc.minAreaRect(new
MatOfPoint2f(contour.toArray()));
            Point[] points = new Point[4];
            rectangle.points(points);

            Point[] sortedPoints = sortPointArray(points);
            double width = sortedPoints[1].x - sortedPoints[0].x;
            double distance = (4537 / width) * 2.54;

            return distance;
      }

}
package com.identification.bezruchenko.utils

import org.opencv.core.Point;

public class Numeric {

      public double[] sumPoints(Point[] corners)
      {
            /*
             * Design for axis value equal to one(1)
             * This function sums the x and y values of a point
             * returns the same length result array(summed)
             * */
            double[] values = new double[corners.length];
```

```
		for(int i=0;i<corners.length;i++)
			values[i] = corners[i].x + corners[i].y;

		return values;
	}

	public double[] diffPoints(Point[] corners)
	{
		/*
		 * Design for axis value equal to one(1)
		 * This function subt. the x and y values of a point
		 * returns the same length result array(subtracted)
		 * */
		double[] values = new double[corners.length];

		for(int i=0; i<corners.length; i++)
			values[i] = corners[i].x - corners[i].y;

		return values;
	}

	public int argMax(double[] values)
	{
		/*
		 * This function finds the index of maximum value in the destination
array and returns.
		 * */
		int index = 0;
		double  maxValue = values[index]; // initialized with the first element
of the destination array.

		for(int i=1;i<values.length;i++)
		{
			if(values[i]>maxValue) // find the maximum value
			{
				maxValue = values[i];
				index = i;
			}
		}

		return index; //returns the index of maximum value in the dst array.
	}

	public int argMin(double[] values)
	{
		/*
		 * This function finds the index of minimum value in the destination
array and returns.
		 * */

		int index = 0;
		double minValue = values[index];// initialized with the first element of
the destination array.

		for(int i=1; i<values.length;i++)
		{
			if(values[i] < minValue) // find the minimum value
			{
				minValue = values[i];
```

```
                        index = i;
                    }
                }

                return index;//returns the index of minimum value in the dst array.
        }

}
package com.identification.bezruchenko.utils
import org.opencv.core.Mat;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;

import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;
import java.awt.image.WritableRaster;

import javax.swing.ImageIcon;

public class Converter {

        public ImageIcon createAwtImage(Mat mat) {

            int type = 0;
            if (mat.channels() == 1)
                type = BufferedImage.TYPE_BYTE_GRAY;
            else if (mat.channels() == 3)
                type = BufferedImage.TYPE_3BYTE_BGR;
            else
              return null;

            BufferedImage image = new BufferedImage(mat.width(), mat.height(), type);
            WritableRaster raster = image.getRaster();
            DataBufferByte dataBuffer = (DataBufferByte) raster.getDataBuffer();
            byte[] data = dataBuffer.getData();
            mat.get(0, 0, data);

            return new ImageIcon(image);
        }
        public Mat resizeImage(Mat mat, int width, int height)
        {
              Mat copy = new Mat();
              mat.copyTo(copy);
              Imgproc.resize(copy, copy, new Size(width, height));

              return copy;
        }

}
package com.identification.bezruchenko.utils

import org.apache.commons.math3.stat.regression.SimpleRegression;
import org.bytedeco.javacpp.indexer.DoubleRawIndexer;
import org.sanstorik.neural_network.face_identifying.FaceFeatures;

import java.net.URL;
import java.util.ArrayList;
import java.util.List;
```

```
import static org.bytedeco.javacpp.flandmark.*;
import static org.bytedeco.javacpp.opencv_core.*;
import static org.bytedeco.javacpp.opencv_imgproc.getRotationMatrix2D;
import static org.bytedeco.javacpp.opencv_imgproc.warpAffine;

class UserFaceAligner {
    private enum landmark_pos {
        FACE_CENTER(0),
        LEFT_EYE_INNER(1),
        RIGHT_EYE_INNER(2),
        MOUTH_LEFT(3),
        MOUTH_RIGHT(4),
        LEFT_EYE_OUTER(5),
        RIGHT_EYE_OUTER(6),
        NOSE_CENTER(7),
        LEFT_EYE_ALIGN(8),
        RIGHT_EYE_ALIGN(9);

        public int pos;

        landmark_pos(int pos) {
            this.pos = pos;
        }
    }

    private static final String LOADER_URL = "save_session/flandmark_model.dat";

    private static UserFaceAligner userFaceAligner;
    private final FLANDMARK_Model flandmarkModel;


    private UserFaceAligner() {
        URL loaderUrl = getClass().getClassLoader().getResource(LOADER_URL);
        if (loaderUrl == null) throw new IllegalStateException("Landmark model not
found.");

        flandmarkModel = flandmark_init(loaderUrl.getPath());
    }

    public MatFace align(Mat image, Rect bounds) {
        List<Point2d> landmarks = getEyesLandmarks(image, bounds);

        if (landmarks == null) {
            System.out.println("No landmarks");
            return null;
        }

        addAlignedEyesPos(landmarks);
        Mat rotatedFace = rotateImageAndExtractFace(image, landmarks,
                UserFaceDetector.INITIAL_WIDTH, UserFaceDetector.INITIAL_HEIGHT);
        int faceType = findFaceType(landmarks);

        return new MatFace(rotatedFace, faceType);
    }

    public Point2d[] getEyesCenterCoordinates(Mat image, Rect bounds) {
        List<Point2d> landmarks = getEyesLandmarks(image, bounds);

        if (landmarks == null) {
```

```
            return null;
        }

        Point2d leftEyeOuter = landmarks.get(landmark_pos.LEFT_EYE_OUTER.pos);
        Point2d leftEyeInner = landmarks.get(landmark_pos.LEFT_EYE_INNER.pos);

        Point2d rightEyeOuter = landmarks.get(landmark_pos.RIGHT_EYE_OUTER.pos);
        Point2d rightEyeInner = landmarks.get(landmark_pos.RIGHT_EYE_INNER.pos);

        return new Point2d[] { center(leftEyeOuter, leftEyeInner),
center(rightEyeOuter, rightEyeInner) };
    }


    private int findFaceType(List<Point2d> landmarks) {
        int type;

        final double center = landmarks.get(landmark_pos.NOSE_CENTER.pos).x();
        final double left_eye = landmarks.get(landmark_pos.LEFT_EYE_INNER.pos).x();
        final double right_eye = landmarks.get(landmark_pos.RIGHT_EYE_INNER.pos).x();

        System.out.println(center + "" + left_eye + "" + right_eye);
        //distances between them
        double left_center = Math.abs(left_eye - center);
        double right_center = Math.abs(center - right_eye);

        right_center = right_center == 0 ? 1e-3 : right_center;
        left_center = left_center == 0 ? 1e-3 : left_center;


        if (landmarks.size() <= 8) {
            throw new IllegalStateException("Not enough landmarks");
        }

        System.out.println("landmarks = " + landmarks.size());
        System.out.println("right = " + right_center / left_center);
        System.out.println("left = " + left_center / right_center);
        //2.2-2.5 is a perfect threshold
        final double scale_threshold = 2.2;
        if (right_center / left_center >= scale_threshold) {
            type = FaceFeatures.RIGHT_FACE;
        } else if ( left_center / right_center >= scale_threshold) {
            type = FaceFeatures.LEFT_FACE;
        } else {
            type = FaceFeatures.CENTER_FACE;
        }

        System.out.println("Found face type = " + type);

        return type;
    }


    private Point2d center(Point2d left, Point2d right) {
        return new Point2d(
                (left.x() + right.x()) * 0.5,
                (left.y() + right.y()) * 0.5
        );
    }
```

```
    private List<Point2d> getEyesLandmarks(Mat image, Rect bounds) {
        int[] sizes = new int[]{ bounds.x(), bounds.y(),
                bounds.x() + bounds.width(), bounds.y() + bounds.height() };

        final double[] landmarksCoord = new double[2 *
flandmarkModel.data().options().M()];

        flandmark_detect(new IplImage(image), sizes, flandmarkModel, landmarksCoord);

        boolean found = false;
        for(int i = 0; i < landmarksCoord.length; i++) {
            if (landmarksCoord[i] != 0.0) {
                found = true;
            }
        }

        //couldn't find any landmarks for eyes
        if (!found) {
            return null;
        }

        List<Point2d> landmarks = new ArrayList<>(5);

        for (int i = 0; i < this.flandmarkModel.data().options().M(); i++) {
            landmarks.add(new Point2d(landmarksCoord[2 * i], landmarksCoord[2 * i +
1]));
        }

        return landmarks;
    }


    private void addAlignedEyesPos(List<Point2d> landmarks) {
        SimpleRegression linearRegression = new SimpleRegression();

        linearRegression.addData(
                landmarks.get(landmark_pos.LEFT_EYE_OUTER.pos).x(),
                landmarks.get(landmark_pos.LEFT_EYE_OUTER.pos).y()
        );

        linearRegression.addData(
                landmarks.get(landmark_pos.LEFT_EYE_INNER.pos).x(),
                landmarks.get(landmark_pos.LEFT_EYE_INNER.pos).y()
        );

        linearRegression.addData(
                landmarks.get(landmark_pos.RIGHT_EYE_INNER.pos).x(),
                landmarks.get(landmark_pos.RIGHT_EYE_INNER.pos).y()
        );

        linearRegression.addData(
                landmarks.get(landmark_pos.RIGHT_EYE_OUTER.pos).x(),
                landmarks.get(landmark_pos.RIGHT_EYE_OUTER.pos).y()
        );
        Point2d alignedLeftEyePos = new Point2d(
                landmarks.get(landmark_pos.LEFT_EYE_OUTER.pos).x(),
```

```
linearRegression.predict(landmarks.get(landmark_pos.LEFT_EYE_OUTER.pos).x())
        );
        Point2d alignedRightEyePos = new Point2d(
                landmarks.get(landmark_pos.RIGHT_EYE_OUTER.pos).x(),

linearRegression.predict(landmarks.get(landmark_pos.RIGHT_EYE_OUTER.pos).x())
        );

        landmarks.add(alignedLeftEyePos);
        landmarks.add(alignedRightEyePos);
    }


    private Mat rotateImageAndExtractFace(Mat image, List<Point2d> landmarks,  int
faceWidth, int faceHeight) {
        final double DESIRED_LEFT_EYE_X = 0.27;
        final double DESIRED_LEFT_EYE_Y = 0.4;

        Point2d leftEye = landmarks.get(landmark_pos.LEFT_EYE_ALIGN.pos);
        Point2d rightEye = landmarks.get(landmark_pos.RIGHT_EYE_ALIGN.pos);

        Point2f eyesCenter = new Point2f(
                (float)((leftEye.x() + rightEye.x()) * 0.5f),
                (float)((leftEye.y() + rightEye.y()) * 0.5f)
        );

        final double dy = (rightEye.y() - leftEye.y());
        final double dx = (rightEye.x() - leftEye.x());
        final double len = Math.sqrt(dx * dx + dy * dy);
        final double angle = Math.atan2(dy, dx) * 180.0 / CV_PI;

        final double DESIRED_RIGHT_EYE_X = 1 - DESIRED_LEFT_EYE_X;
        final double desiredLen = (DESIRED_RIGHT_EYE_X - DESIRED_LEFT_EYE_X) *
faceWidth;
        double scale = desiredLen / len;

        Mat rotMat = getRotationMatrix2D(eyesCenter, angle, scale);
        DoubleRawIndexer indexer = rotMat.createIndexer();
        indexer.put(0, 2, indexer.get(0, 2) + (faceWidth * 0.5 - eyesCenter.x()));
        indexer.put(1, 2, indexer.get(1, 2) + (faceHeight * DESIRED_LEFT_EYE_Y -
eyesCenter.y()));

        Mat destImage = new Mat(faceHeight, faceWidth, CV_8U, new Scalar(128));
        warpAffine(image, destImage, rotMat, destImage.size());
        return destImage;
    }


    public static UserFaceAligner create() {
        if (userFaceAligner == null) {
            userFaceAligner = new UserFaceAligner();
        }

        return userFaceAligner;
    }
}
```