

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ІНТЕЛЕКТУАЛЬНА СИСТЕМА ІДЕНТИФІКАЦІЇ
ЛЮДИНИ ЗА ФОТО

Виконав: студент 2 курсу, групи КНТ-512м

Спеціальності 123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
«Комп'ютерні системи та мережі»

БОБИРЬ Б.М.

(ПРИЗВИЩЕ та ініціали)

Керівник ТЯГУНОВА М.Ю

(ПРИЗВИЩЕ та ініціали)

Рецензент СФИМЕНКО М.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
 Кафедра «Комп'ютерні системи та мережі»
 Ступінь вищої освіти магістерський
 Спеціальність 123 Комп'ютерна інженерія
 (код і найменування)
 Освітня програма (спеціалізація) Комп'ютерні системи та мережі
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“ ” 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

БОБИРЯ Богдана Миколайовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Інтелектуальна система ідентифікації людини за фото

керівник проєкту (роботи) к. т. н., доцент, ТЯГУНОВА Марія Юріївна

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “24” жовтня 2023 р. №400

2. Строк подання студентом проєкту (роботи) 10 грудня 2023 року

3. Вихідні дані до проєкту (роботи) наявні методи та алгоритми пошуку облич на зображні та відео, швидкий пошук орієнтирів обличчя, ефективний пошук по базі даних збережених облич

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз методів та інструментів розробки системи ідентифікації;

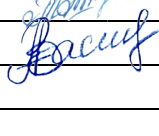
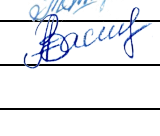
2) Проєктування системи;

3) Реалізація системи;

4) Результати дослідження.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ТЯГУНОВА М. Ю., доцент		
нормоконтроль	ПОЛЬСЬКА О.В., ст.викл.		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз наявних методів та алгоритмів пошуку об'єктів, перегляд отриманих результатів, виділення	10.10.2023 р.	
	недоліків та переваг		
2	Розробка структури системи	15.10.2023 р.	
3	Розробка алгоритму роботи системи	20.10.2023 р.	
4	Реалізація системи уточненого розпізнавання	05.10.2023 р.	
5	Дослідження системи уточненого розпізнавання	20.11.2023 р.	
6	Оформлення отриманих результатів у ПЗ	25.11.2023 р.	
7	Оформлення графічного матеріалу	01.12.2023 р.	
8	Оформлення допоміжного матеріалу	10.12.2023 р.	

Студент

 Богдан БОБИРЬ
(підпис) (ім'я, ПРИЗВИЩЕ)

Керівник проєкту (роботи)

 Марія ТЯГУНОВА
(підпис) (ім'я, ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 75 с., 49 рис., 1 табл., 18 джерел.

РОЗПІЗНАВАННЯ ОБЛИЧЧЯ, ІДЕНТИФІКАЦІЯ ОБЛИЧЧЯ, ГЛИБОКЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, OPENCV, DLIB, GPU

Об'єкт дослідження – системи ідентифікації людини на фото.

Предмет дослідження – алгоритми глибокого навчання та апаратні засоби для ідентифікації людини на фото.

Мета роботи – дослідити методи розпізнавання обличчя за фото та розробити систему, за допомогою якої можливо ідентифікувати довільне обличчя із вже збережених раніше.

У першому розділі розглянуто основні методи пошуку обличчя на окремо взятому зображенні, а також проведено аналіз апаратних рішень для забезпечення ефективної роботи алгоритмів пошуку, таких як GPU та TPU.

У другому розділі спроектовано систему, розглянуто важливість використання бібліотек OpenCV і Dlib та інтерфейс взаємодії з графічною картою CUDA, а також наведено обрані алгоритми для етапів розпізнавання обличчя. Проаналізовано PostgreSQL та MySQL для використання в системі.

У третьому розділі було розглянуто практичне застосування використаних в системі бібліотек DLib, Opencv, Tensorflow та Mediarpipe, проведено порівняння швидкості роботи системи при розпізнаванні фото з 1, 5 та 10 обличчями, побудовано графік для кожного підходу і кожної умови та на протязі 100 ітерацій. Наведено ключові дані реалізованих серверного та клієнтського застосунків.

У четвертому розділі наведено результати роботи взаємодії серверного та клієнтського застосунків, протестовано їх роботу на прикладі декількох тестових завдань, та проаналізовано отримані результати.

ABSTRACT

MT: 75 pp., 49 fig., 1 tables, 18 sources.

FACE RECOGNITION, FACE IDENTIFICATION, DEEP LEARNING,
NEURAL NETWORKS, OPENCV, DLIB, GPU

The object of the research is the identification of the person in the photo.

The subject of research is a combination of deep learning algorithms and hardware solutions.

The goal of the research is to develop a system that can identify an arbitrary face from previously saved ones.

In the first section, the main methods of face detection in a single image are considered, and hardware solutions for efficient operation of search algorithms, such as GPU and TPU, are also analyzed.

In the second section, the system is designed, the importance of using the OpenCV and Dlib libraries and the interface of interaction with the CUDA graphics card are considered, and the selected algorithms for the stages of face recognition are given. Analyzed PostgreSQL and MySQL for use in the system.

In the third section, the practical application of the DLib, Opencv, Tensorflow and Mediapipe libraries used in the system was considered, a comparison was made of the speed of the system when recognizing photos with 1, 5 and 10 faces, a graph was drawn for each approach and each condition and over 100 iterations. The key data of implemented server and client applications are given.

In the fourth section, the results of the interaction of server and client applications are presented, their operation is tested on the example of several test tasks, and the obtained results are analyzed.

ЗМІСТ

Вступ.....	7
1 Аналіз методів та інструментів розробки системи ідентифікації.....	9
1.1 Методи виділення обличчя.....	10
1.2 Методи розпізнавання орієнтирів обличчя.....	17
1.3 Методи розпізнавання особи за обличчям.....	22
1.4 Аналіз апаратних рішень.....	23
1.5 Висновки до розділу 1.....	28
2 Проєктування системи.....	29
2.1 Загальний опис системи.....	29
2.2 Вибір архітектури системи.....	30
2.3 Розробка структури системи.....	34
2.4 Функціонування системи.....	37
2.5 Висновки до розділу 2.....	45
3 Реалізація системи.....	46
3.1 Визначення навчальних та тестових вибірок.....	46
3.2 Реалізація модулів системи.....	55
3.3 Висновки до розділу 3.....	60
4 Результати дослідження.....	61
4.1 Тестування системи.....	62
4.2 Аналіз отриманих результатів.....	67
4.3 Висновки до розділу 4.....	71
Висновки.....	72
Перелік джерел посилання.....	74

ВСТУП

Усі досягнення в технічному прогресі людства базуються на відкритті нових технологій, а потім спрощенні їхньої реалізації. У кожній сфері нашого життя постійно відбувається розвиток, що призводить до виникнення кращих рішень для вирішення завдань та проблем. Інформаційні технології не є винятком і також піддаються постійному вдосконаленню.

Одним з таких рішень, яке зараз набуло дуже широкого використання та великої популярності є нейронні мережі. Нейронні мережі (або штучні нейронні мережі) – це клас алгоритмів та моделей машинного навчання, структура та будова яких навіяна будовою біологічної нейронної системи та мозку.

Нейронні мережі виникли як технологія, покликана трансформувати світ, та яка вже революціонізувала різні сфери завдяки своїй глибокій важливості та впливу. Моделі штучного інтелекту, створені за допомогою нейронної структури людського мозку, започаткували нову еру інновацій, сприяючи прогресу в різних галузях.

Будучи піонером в вирішенні складних завдань, нейронні мережі чудово справляються зі складними проблемами в різних областях. Їхня здатність вивчати закономірності, взаємозв'язки та шаблони в даних робить їх корисними в таких завданнях, як розпізнавання зображень і мови, обробка природної мови, прогнозна аналітика та прийняття рішень.

Однією з сильних сторін нейронних мереж є розпізнавання образів. Ідентифікація об'єктів на зображеннях, трансформування мови в текст або розуміння почуттів у тексті – нейронні мережі ефективно розпізнають та інтерпретують складні шаблони з величезних наборів даних. В свою чергу, ідентифікація об'єктів та розпізнавання образів має широке застосування в багатьох галузях, таких як, наприклад, автономні системи - в автономних транспортних засобах і дронах розпізнавання об'єктів дозволяє виявляти

пішоходів, транспортні засоби, дорожні знаки та перешкоди. Це підвищує безпеку, дозволяючи цим системам приймати рішення в реальному часі на основі ідентифікованих об'єктів в їх оточенні. Або в галузі безпеки - у системах відеоспостереження розпізнавання об'єктів допомагає ідентифікувати та відстежувати об'єкти чи осіб, що дає змогу забезпечувати безпеку, виявляючи підозрілі дії або розпізнаючи обличчя в громадських місцях. Таким чином, можна зробити висновок, що обрана тема магістерської роботи є більш ніж актуальною.

Мета роботи – дослідити методи розпізнавання обличчя за фото та розробити систему, за допомогою якої можливо ідентифікувати довільне обличчя із вже збережених раніше.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- проаналізувати методи та інструменти, необхідні для розробки систем ідентифікації облич;
- спроектувати систему ідентифікації та описати її функціонування;
- визначити навчальну та тестову вибірки для навчання системи;
- реалізувати модулі системи та систему в цілому;
- провести тестування системи та проаналізувати отримані результати.

1 АНАЛІЗ МЕТОДІВ ТА ІНСТРУМЕНТІВ РОЗРОБКИ СИСТЕМИ ІДЕНТИФІКАЦІЇ

Зважаючи на те, що цей проєкт базується на теорії розпізнавання образів, важливо спочатку ознайомитися з основними принципами цієї теорії та науковою літературою, яка детально розглядає цей напрямок.

Теорія розпізнавання образів застосовується у різних галузях, а не лише в інформаційних технологіях. Її основна ідея полягає в ідентифікації та класифікації різних об'єктів, процесів, сигналів і явищ на основі їх характеристик. Останнім часом теорія розпізнавання образів набула великої популярності в галузі штучного інтелекту, і зацікавленість цим питанням постійно зростає.

Вперше ця теорія отримала свій розвиток наприкінці 50-х років минулого століття. Основною метою було створення механізму, який міг би класифікувати образи та ситуації, подібно до того, як це робить жива людина. З часом це питання стало дуже обширним, і теорія розпізнавання образів розділилася на кілька напрямків, кожен з яких спрямовувався на вирішення більш конкретних завдань у різних галузях.

Одним з таких напрямків є виділення об'єкта на відеопотоках в прикладних задачах. Необхідність визначення об'єкта на відеопотоках в прикладних задачах може бути у визначенні небажаних осіб чи віп клієнтів публічних місцях, зібранні статистичні дані для подальшого аналізу, наприклад визначення в який день приходить найбільше відвідувачів, медичних установах відеоспостереження може використовуватися для контролю доступу до обмежених зон, відстежування пацієнтів та співробітників, а також для забезпечення безпеки та дотримання правил установи тощо.

У цій дипломній роботі ми зосередились на виділенні облич людей та подальшій їх обробці. Також буде глибоко досліджено різні алгоритми,

придатні для виявлення об'єктів. Серед них представлено алгоритми, що входять до сімейства RCNN, такі як RCNN, Fast RCNN і Faster RCNN. Крім цього, розглянуто більш вдосконалені алгоритми, такі як YOLO, SSD і інші. Для реалізації цих алгоритмів використаємо бібліотеку OpenCV, фреймворк Tensorflow, DLib, Mediapipe та інші [1].

1.1 Методи виділення обличчя

Важливим етапом проєктування системи ідентифікації людини на зображенні є базове виділення облич на зображенні (приклад зображено на рисунку 1.1), тобто пошук всіх облич на фото або відео, конвертованим у масив фото. Зазвичай це передбачає використання комп'ютером алгоритму машинного навчання для ідентифікації певних особливостей зображення, наприклад, форми обличчя, положення носу та очей тощо. Коли обличчя та його основні компоненти виявлено, їх можна відстежувати та аналізувати в подальшому наприклад, для забезпечення безпеки.



Рисунок 1.1 – Демонстрація роботи визначення осіб на зображенні

Розпізнавання обличчя є активною сферою досліджень у галузі комп'ютерного зору, і протягом багатьох років було розроблено багато різних підходів. Тож розглянемо найпопулярніші з них (рисунок 1.2).



Рисунок 1.2 – Підходи виділення об'єктів на зображенні

Метод каскадних класифікаторів (Cascade Classifiers) - метод на основі знаходження областей інтересу (ROI), використовує попередньо навчені класифікатори для швидкого відсіювання негативних областей та дозволяє швидко визначити регіони, які мають потенційно містити обличчя. Одним із популярних методів є використання класифікаторів Хаара[2] (приклад роботи зображено на рисунку 1.3) або ЛБП (локальні бінарні шаблони).

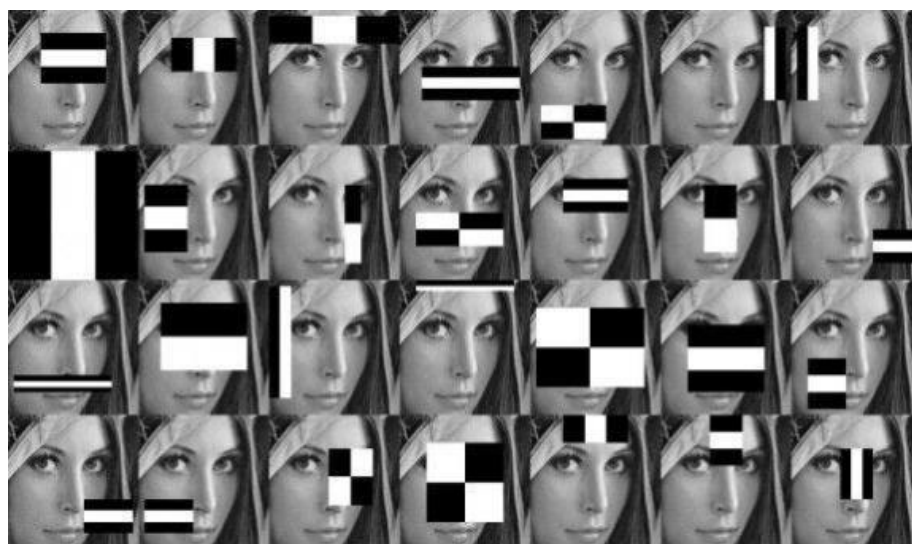


Рисунок 1.3 – Застосування каскадів Хаари для виявлення обличчя

Каскади Хаара (Haar Cascades) - це метод для виявлення об'єктів, розроблений В'ячеславом Хааром і Пауль Гебеке у 2001 році. Цей метод використовує навчені класифікатори, які використовуються для виявлення об'єктів на зображеннях або відеопотоках. Каскади Хаара широко використовуються для виявлення облич, але їх також можна навчити для виявлення інших об'єктів.

Основна ідея каскадів Хаара полягає в швидкому фільтруванні зображення шляхом послідовної перевірки признаков. Каскад складається з багатьох етапів (класифікаторів), кожен з яких застосовується послідовно. Якщо під час будь-якого етапу знайдено відсутність об'єкта, то обробка припиняється, і зображення відкидається. Це дозволяє швидко відсіяти області без об'єктів, що значно прискорює процес виявлення.

Основним недоліком каскадів є те, що у порівнянні з моделями глибокого навчання, каскади Хаара зазвичай менш точні та менш універсальні. З переваг, це те, що вони мають перевагу у швидкодії та вимагають менше обчислювальних ресурсів.

Перед тим як дати визначення згортковим нейронним мережам, потрібно з'ясувати, що таке нейронні мережі. Нейронна мережа – це метод штучного інтелекту, а також набір алгоритмів, який намагається розпізнати базові зв'язки в наборі даних за допомогою процесу, який імітує роботу людського мозку.. Це тип процесу машинного навчання, який називається глибоким навчанням, який використовує взаємопов'язані вузли або нейрони в багат шаровій структурі, що нагадує людський мозок. Він створює адаптивну систему, за допомогою якої комп'ютери навчаються на своїх помилках і постійно вдосконалюються. Таким чином, штучні нейронні мережі намагаються вирішувати складні проблеми, такі як узагальнення документів або розпізнавання облич, з більшою точністю.

Перші кроки до створення штучних нейронних мереж були зроблені у далекому 1943 році, коли Уоррен МакКалох, нейрофізіолог, і молодий математик Уолтер Пітс написали статтю про те, як можуть працювати нейрони. Вони змоделювали просту нейронну мережу з електричними схемами.

У 1950-х роках робота Розенблата призвела до створення двошарової мережі, персептрона, яка була здатна вивчати певні класифікації шляхом регулювання ваги з'єднань, але також мала деякі обмеження. З початку 1980-х років і до сьогодні інтерес до нейронних мереж тільки зростає, а кількість їх видів збільшується з кожним роком.

Оскільки людський мозок є джерелом натхнення для архітектури нейронної мережі, де клітини людського мозку, які називаються нейронами, утворюють складну взаємопов'язану мережу та посиляють одна однієї електричні сигнали, допомагаючи людям обробляти інформацію, то так само і штучна нейронна мережа складається зі штучних нейронів, які працюють разом, щоб вирішити проблему. Штучні нейрони, або перцептрони, це програмні модулі, які називаються вузлами, а штучні нейронні мережі – це програмні програми або алгоритми, які, за своєю суттю, використовують обчислювальні системи для виконання математичних обчислень.

Найвідомішим та самим простим прикладом нейронної мережі є багатошаровий персептрон. У багатошаровому перцептроні (MLP) нейрони розташовані у взаємопов'язаних шарах. Вхідний рівень збирає вхідні шаблони або патерни. Вихідний рівень має класифікації або вихідні сигнали, які є результатом роботи, і асоціюються з вхідними даними. Наприклад, якщо шаблони можуть містити список величин для технічних індикаторів цінних паперів; то потенційні результати можуть бути «купити», «тримати» або «продавати». Будова простого багатошарового перцептрону показана на рисунку 1.4.

Окрім багатошарового перцептрону існує багато інших, більш складних нейронних мереж, які зазвичай описуються кількістю прихованих вузлів, або або кількістю вхідних і вихідних шарів у кожному вузлі, також є різні форми розповсюдження інформації, наприклад пряма (feed-forward) та зворотня (backward) форми. Наприклад, багатошаровий перцептрон належить до нейронних мереж прямого зв'язку (Feed-forward neural network).

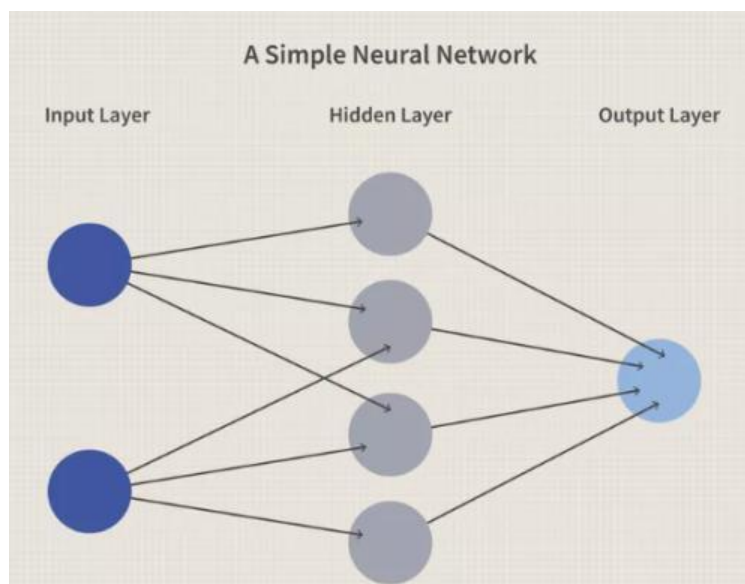


Рисунок 1.4 – Багатошаровий перцептрон

Більш складними за будовою є повторювані нейронні мережі (Recurrent neural networks, RNN)[3], зберігають вихідні дані вузлів обробки та повертають результат назад у модель. Так модель вчиться передбачати результат шару. Кожен вузол у моделі RNN діє як комірка пам'яті, продовжуючи обчислення та виконання операцій.

Ця нейронна мережа починається з того самого фронту поширення, що й мережа прямого зв'язку, але потім продовжує запам'ятовувати всю оброблену інформацію для повторного використання в майбутньому. Якщо передбачення мережі є неправильним, то система самонавчається та продовжує працювати над правильним передбаченням під час зворотного поширення. Цей тип нейронних мереж часто використовується для перетворення тексту на мовлення.

Наступним популярним методом виділення об'єктів та облич на зображенні є згорткові нейронні мережі (Convolutional Neural Network) - це тип нейронних мереж глибокого навчання, який особливо підходить для обробки та аналізу зображень, і став основою багатьох успішних застосувань в галузі комп'ютерного зору, включаючи розпізнавання облич, класифікацію зображень, виявлення об'єктів тощо. CNN дозволяє автоматично вивчати ієрархічні та локальні функції зображень, забезпечуючи ефективне виявлення інформації в

різних рівнях абстракції. Типова згорткова мережа складається з великої кількості шарів:

- згорткові шари (Convolutional Layers) – використовують фільтри для здійснення операції згортки з вхідним зображенням, що дозволяє нейронам виявляти локальні особливості, такі як границі та форми;
- шари пулінгу (Pooling Layers) – шари зменшують розмір виходу попередніх шарів. Це полегшує обробку та зменшує кількість параметрів мережі;
- повністю з'єднані шари (Fully Connected Layers) – шари використовуються для розділення характеристик, вивчених попередніми шарами, для класифікації чи виявлення.

Приклад типової структури згорткової мережі зображено на рисунку 1.5

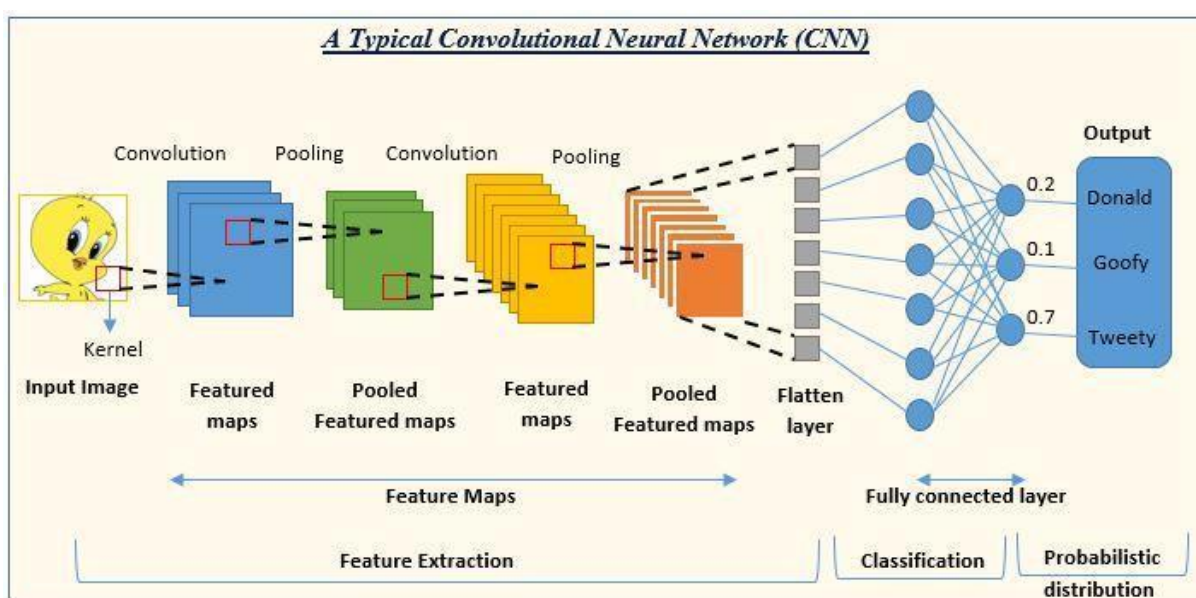


Рисунок 1.5 – Структура згорткової нейронної мережі

Методи виявлення контурів та геометричних форм – різноманітні математичні методи, спрямовані на виявлення контурів та кривих на зображенні, на яких яскравість зображення змінюється різко або має розриви.

В ідеальному випадку результатом застосування до зображення виявлення контурів може бути набір з'єднаних кривих, що позначають межі

об'єктів, межі забарвлення поверхонь, а також усі криві, що відповідають розривам у спрямуванні поверхонь. Таким чином, застосування алгоритму виявлення контурів до зображення може значно зменшувати кількість даних, що підлягають обробці, й відтак може відфільтровувати інформацію, яку можна розцінювати як менш значущу, зберігаючи при цьому важливі структурні властивості зображення.

Розглянемо деякі приклади методів виділення контурів на зображенні. Оскільки їх існує дуже велика кількість, перелічимо найпопулярніші алгоритми:

- алгоритм Кенні – використовується для виявлення границь та контурів на зображенні та включає в себе етапи виявлення горизонтальних та вертикальних границь, виявлення границь за допомогою градієнта, а також виявлення контурів за допомогою операції не максимального здавлювання (non-maximum suppression) Приклад роботи алгоритму Кенні зображено на рисунку 1.6;

- алгоритм Собеля - це простий алгоритм виявлення границь, який обчислює градієнт інтенсивності зображення. Застосовуючи алгоритм Собеля як у горизонтальному, так і у вертикальному напрямках, можна виявити границі та окреслити контури. Приклад роботи алгоритму Собеля зображено на рисунку 1.7;

- алгоритм контуру опуклості (Convex Hull) - використовується для побудови опуклої оболонки навколо набору точок на зображенні, чим дозволяє виявити форму та контур об'єкта;

- алгоритм Рамера-Дугласа-Пекера (Ramer-Douglas-Peucker) - це рекурсивний алгоритм, який використовується для спрощення лінії або кривої на площині, зберігаючи її загальну форму. Особливо корисний у випадках, коли потрібно зменшити кількість точок, які описують лінію, з мінімальною втратою інформації.

Ці методи та алгоритми можуть використовуватися окремо або комбінуватися в залежності від конкретного завдання виявлення контурів та геометричних форм на зображенні.



Рисунок 1.6 – Застосування алгоритму Кенні до зображення

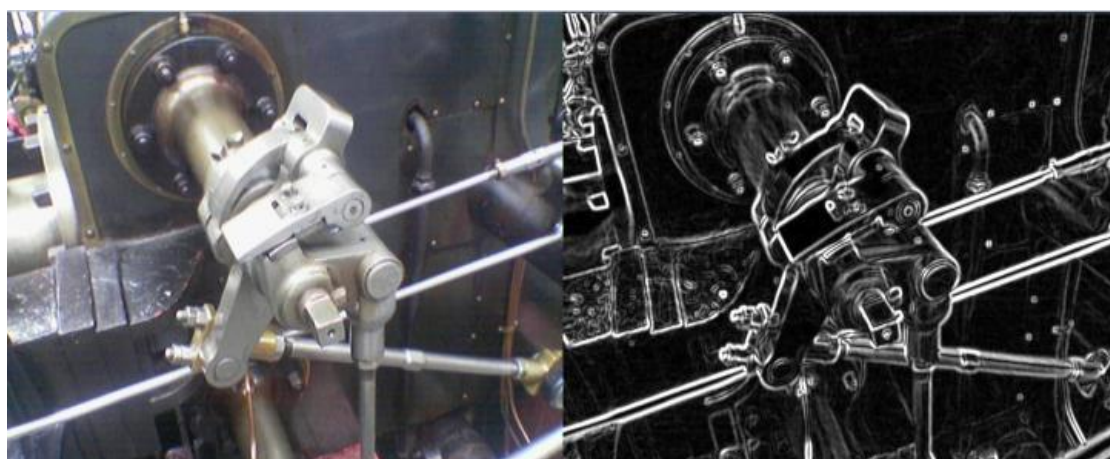


Рисунок 1.7 – Застосування алгоритму Собеля до зображення

1.2 Методи розпізнавання орієнтирів обличчя

Для початку потрібно дати визначення терміну орієнтири обличчя. Орієнтири обличчя – це конкретні точки або риси на обличчі людини, які служать опорними точками (зображено на рисунку 1.8) для різних застосувань у комп'ютерному зорі, аналізі та відстеженні обличчя та емоцій, а також в

суміжних областях. Орієнтири обличчя відіграють вирішальну роль у різних програмах, таких як розпізнавання обличчя, аналіз емоцій і визначення виразу обличчя. Аналізуючи положення та відстані між цими ключовими точками, алгоритми можуть характеризувати риси обличчя та робити висновки про особистість, емоції та інші атрибути обличчя.

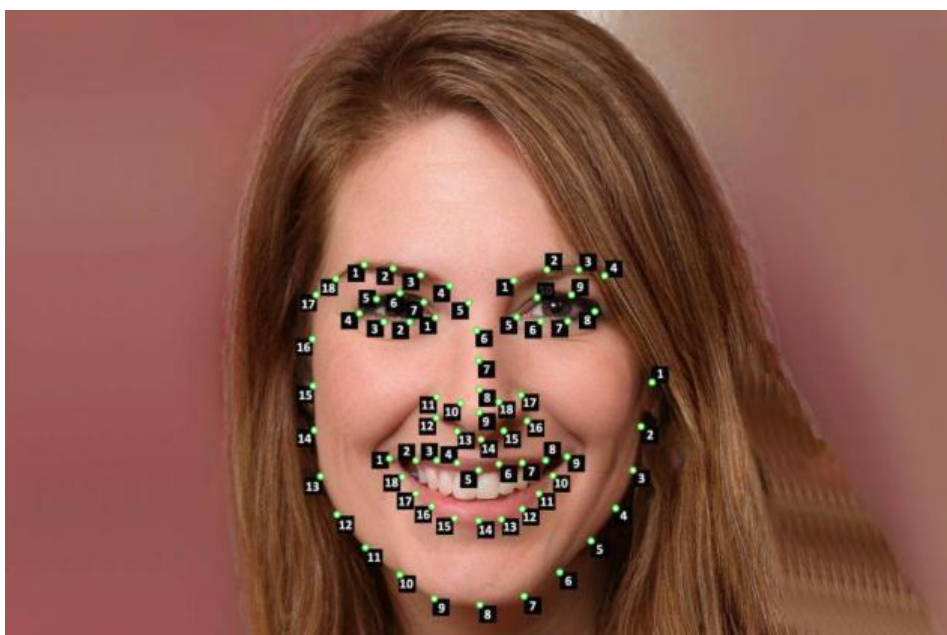


Рисунок 1.8 – Приклад опорних точок на обличчі

Надалі ці детальні орієнтири обличчя можуть бути використані для створення унікального шаблону обличчя для ідентифікації осіб, який, в свою чергу, можна зберегти і використовувати надалі - тобто для системи людина буде відома як пара ім'я - шаблон. Розглянемо більш детально як проходить процес створення унікального шаблону. Оскільки сам процес створення - це маніпуляція з координатами точок, які були знайдені кроком вище за допомогою нейронних мереж, то можна виділити основні етапи обробки цих точок:

- збір координат;
- нормалізація координат;
- створення числового шаблону.

Спочатку необхідно зібрати координати орієнтирів на обличчі людини за допомогою спеціалізованих моделей глибокого навчання або нейронних мереж. Ці координати включають в себе положення очей, носа, рота, бровей та інших ключових точок. Далі необхідно зробити нормалізацію координат. Зазвичай, координати орієнтирів обличчя нормалізуються для забезпечення сталого масштабу та положення обличчя на зображенні. Це може включати в себе перетворення координат таким чином, щоб обличчя завжди розташовувалося в однаковому положенні та масштабі. Нормалізовані координати орієнтирів обличчя перетворюються в числовий шаблон, який може бути представлений у вигляді вектора чисел. Цей вектор може бути результатом конкатенації всіх координат, наприклад. Інші методи перетворення також можуть використовуватися, в залежності від конкретного алгоритму та системи розпізнавання.

Як приклад числового шаблону обличчя можна показати шаблон як вектор чисел, наприклад:

.25,0.35,0.45,0.55,0.65,0.75,0.85,0.95,1.0.

У цьому прикладі, вектор містить 9 чисел, які відповідають за положення ключових точок на обличчі. Кожне число представляє відсоткове відношення відносно розміру обличчя. Наприклад, перше число 0.25 вказує на положення лівого кута обличчя, а останнє число 1.0 вказує на положення правого кута обличчя. Звичайно, це дещо спрощений приклад, реальний числовий шаблон обличчя може бути значно більшим і складнішим, і включати в себе координати десятків або навіть сотень точок на обличчі. Кожне число може відображати певну анатомічну особливість або положення на обличчі.

Розглянемо деякі популярні алгоритми розпізнавання орієнтирів на обличчі. Багато готових рішень та бібліотек базується на моделі Каземі або Cascaded Shape Regression, CSR - метод на основі регресії, який зазвичай використовується для оцінки положення ключових точок або орієнтирів на об'єкті, як правило, на обличчях людей. Цей метод використовує каскадний підхід, де кожен наступний регресор вдосконалює результуючий вектор

ключових точок. На кожному етапі регресор навчений мінімізувати різницю між його прогнозованою формою та основною істинною формою. Регресори зазвичай використовують набір характеристик, отриманих із вхідного зображення або поточної оцінки форми. Ці ознаки вибрано так, щоб вони були інформативними для завдання локалізації орієнтирів. Приклад роботи CSR показано на рисунку 1.9.

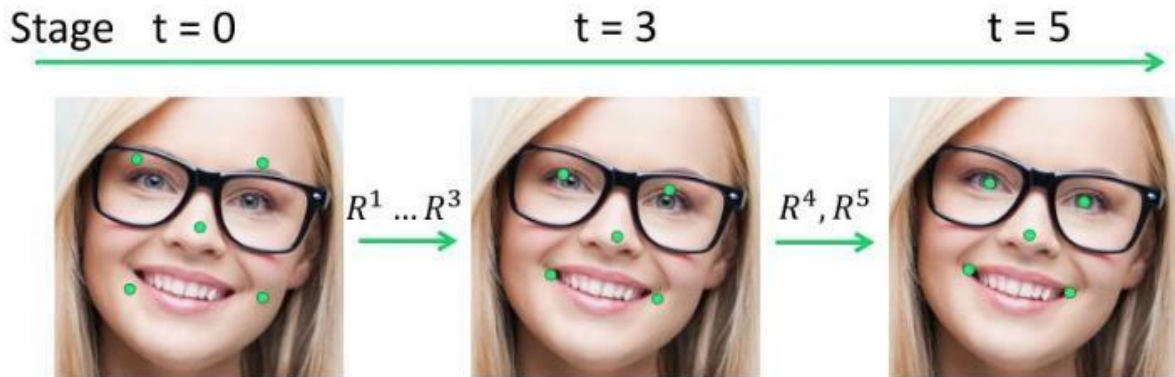


Рисунок 1.9 – Приклад роботи каскадів в CSR

Модель Каземі навчена на великій кількості зображень з правильними координатами ключових точок обличчя, що дозволяє їй точно визначати ці точки на нових зображеннях.

Це одним популярним підходом є обмежена локальна модель (Constrained Local Model, CLM) - підхід в машинному навчанні або обробці сигналів, коли модель або алгоритм фокусується на обробці обмежених, локальних аспектів даних, замість глобального аналізу всього набору даних. Дана модель концентрується на областях чи частинах даних, що може полегшити аналіз великих обсягів інформації, тому може бути більш ефективною з точки зору обчислень, оскільки необхідно опрацьовувати менше даних. Такий підхід може бути більш масштабованим, оскільки дозволяє розглядати окремі частини даних паралельно (рисунок 1.10).

Прикладами можуть бути методи обробки зображень, де для визначення обличчя або об'єктів використовуються локальні фільтри чи обмежені області для виявлення важливих особливостей (рисунок 1.11).

Останній підхід, який ми розглянемо є використання методів, основаних на холістичній (цілісній) моделі - це тип моделі або підходу, який розглядає систему або об'єкт як цілісну сутність, де кожна частина взаємодіє з іншими частинами для створення загального явища або поведінки [4].

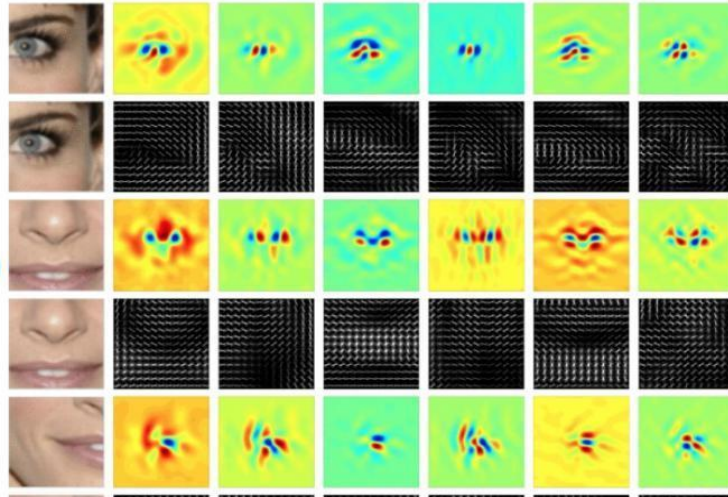


Рисунок 1.10 – Градієнти локального детектора для кутів очей, носа та рота

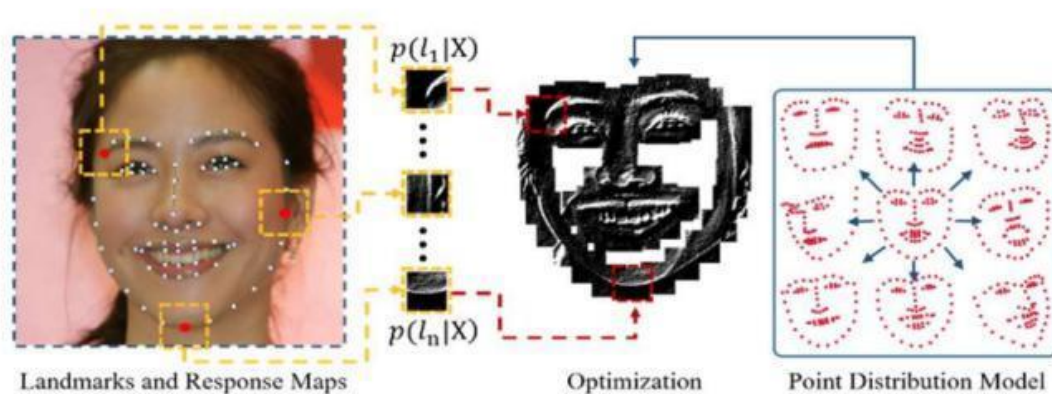


Рисунок 1.11 – Приклад роботи CLM

У холістичних моделях аналізується взаємодія між всіма складовими частинами системи, а не окремо кожна частина відокремлено. Якщо порівнювати з попереднім підходом, то холістична модель є повною протилежністю - на прикладі виділення обличчя модель оброблює відразу все обличчя на відміну від локальної моделі, де ведеться пошук окремі об'єктів лица, такі як очі, брови, губи тощо.

1.3 Методи розпізнавання особи за обличчям

Останнім етапом ідентифікації є порівняння контрольних точок для виявлення співпадінь. Передумовою виявлення співпадінь є, звичайно, накопичена база вже ідентифікованих людей, збережених у форматі, зручному для пошуку. Можна виділити наступний алгоритм:

- на зображенні за допомогою описаних вище етапів знаходяться всі обличчя;
- на кожному обличчі знаходяться контрольні точки та кодуються в векторний формат;
- кожен нормалізований вектор порівнюється з наявною базою, в якій збережені вектор, відповідний лейбл, який ідентифікує людину (рисунок 1.12) та унікальний id;
- якщо людина не знайдена в базі, то вона стає в чергу на ідентифікацію, а поки цього не сталося, буде відображатися її id по типу id123456.

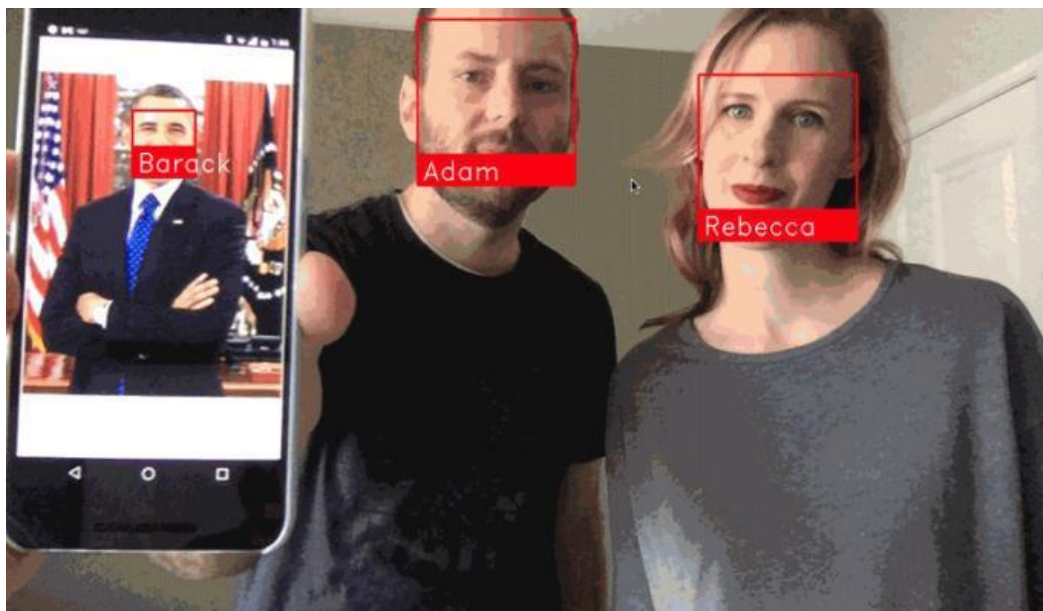


Рисунок 1.12 – Успішна ідентифікація

1.4 Аналіз апаратних рішень

Оскільки розпізнавання образів є задача нетривіальна і вимагає особливих умов роботи, таких як типи процесорів для обробки, бібліотеки та моделі нейронних мереж, розглянемо особливості роботи нейронних мереж на різних типах процесорів:

- CPU – це загально призначений процесор, який використовується для виконання широкого спектру обчислювальних завдань, підходить для невеликих нейронних мереж, але не має великої кількості обчислювальних одиниць, тому він може бути повільним для великих та складних нейронних мереж;

- GPU – графічний процесор, відомий своєю здатністю виконувати паралельні обчислення і підходить для тренування складних та великих нейронних мереж, а також для визначення результатів вимогливих до продуктивності моделей;

- NPU – процесор нейронних мереж, спеціалізується на обчисленнях, пов'язаних з нейронними мережами, і призначений для швидкого виконання операцій над тензорами та матрицями. Він ідеально підходить для нейронних мереж та глибокого навчання.

Оскільки CPU не дуже часто використовують саме для навчання або роботи нейронних мереж (навіть дуже потужні по типу Intel i9 або AMD ThreadRipper), то розглядати їх як апаратне рішення для запуску системи ідентифікації буде неправильно. Але якщо комбінувати CPU з GPU або NPU/TPU, де сама нейронна мережа буде працювати на ефективних GPU або NPU, а наприклад сервіс комунікації, по типу, наприклад, серверу, буде працювати на CPU, то це виглядає як досить непогана ідея. Тому в даному випадку будемо порівнювати ефективність роботи нейронних мереж саме на GPU та NPU/TPU.

Блоки обробки графіки (GPU), спочатку призначені для візуалізації графіки, стали популярним вибором для завдань паралельної обробки. Вони складаються з тисяч невеликих ядер, оптимізованих для обробки векторних і матричних операцій, що робить їх ідеальними для глибокого навчання та інших інтенсивних обчислювальних навантажень[5].

Блоки обробки тензорів (TPU), спеціальні інтегральні схеми (ASIC), розроблені спеціально для завдань машинного навчання. Представлені Google, TPU розроблені для виконання тензорних операцій, які є основними будівельними блоками нейронних мережових обчислень.

У той час як графічні процесори використовують гнучку архітектуру загального призначення, TPU спеціально створені для завдань машинного навчання. Графічні процесори складаються з тисяч маленьких ядер, розроблених для одночасного виконання кількох завдань, тоді як TPU мають більш оптимізовану архітектуру, орієнтовану на прискорення тензорних операцій.

Коли мова заходить про продуктивність, у певних сценаріях TPU мають перевагу над GPU. TPU призначені для виконання обчислень з нижчою точністю з більшою пропускну здатністю, якої часто достатньо для завдань навчання та логічного висновку в нейронних мережах (наприклад точність int8 та FP16). Однак графічні процесори пропонують більшу гнучкість з точки зору точності та можуть обробляти високоточні обчислення, коли це необхідно.

TPU зазвичай мають вищу пропускну здатність пам'яті, ніж GPU, що дозволяє їм більш ефективно обробляти великі тензорні операції. Це призводить до швидкого навчання та часу висновку для нейронних мереж. Однак обсяг пам'яті, доступний на TPU, зазвичай менший, ніж на GPU, що може бути обмежуючим фактором для деяких програм.

Якщо розглядати приклади популярних GPU, то безперечним лідером серед виробників є NVidia, та їх лінійки графічних карт RTX серії 30*/40* та серверні графічні карти A100. На рисунку 1.13 показано порівняння цих GPU. Особливо нас цікавить кількість CUDA (Compute Unified Device Architecture)

ядер – це платформа розробки від NVIDIA, яка дозволяє використовувати графічні процесори в паралельних обчислювальних задачах, а не тільки в графічних операціях. Саме ці ядра, разом з Tensor ядрами, будуть відповідати за швидкодію нейронних мереж.

В якості готового рішення з NPU/TPU процесором можна звернути увагу на такі single-board комп'ютери, як Google Coral (показано на рисунку 1.14) на базі чипу від Google під назвою Edge TPU, та Nvidia Jetson Nano.

	2080ti (Turing)	3060ti (Ampere)	3070 (Ampere)	3080 (Ampere)	3080ti (Ampere)	3090 (Ampere)	A100 (Ampere)
CUDA cores	4352	4864	5888	8704	10240	10496	6912
TensorCores	544		?	?	?	?	?
Memory	11GB	8GB	8GB	10GB	12GB	24GB	40GB
GPU Clock Rate/Boost	1.5/1.63GHz	1.41/1.67	1.5/1.73GHz	1.44/1.71GHz	1.37/1.67	1.4/1.7GHz	765/1410GHz
Memory Bandwidth	616GB/s	448GB/s	448GB/s	760GB/s	912GB/s	936GB/s	1555GB/s
Thermal Rating TDP	250W	200W	220W	320W	350W	350W	250W (PCI-E)/400W (HGX)
FP16 Perf.	114 TFLOPS	~140TFLOPS	163 TFLOPS	238 TFLOPS	273 TFLOPS	285 TFLOPS	312 TFLOPS
Price	SRP: 1200\$	399\$	499\$	699\$	1,199\$	1,499\$	\$9,895

Рисунок 1.13 – Порівняння GPU від Nvidia



NVIDIA Jetson Nano



Google Coral Edge TPU

Рисунок 1.14 – Приклад single-board комп'ютерів з TPU

Дані пристрої мають дуже велику продуктивність, яка вимірюється в трильйонах операцій на секунду (TOPS) та для, наприклад, Google Coral ця

цифра становить 4 TOPS, і при цьому цей комп'ютер споживає 5-10 ватт електроенергії в залежності від завдань та навантажень.

Разом з продуктивністю також потрібно дослідити точність обчислень. Загалом правило таке: чим менше розмір типу обчислювальних даних (16-bit floating-point, 32-bit floating-point, та 8-bit integer, або FP16, FP32 та INT8) тим швидше обчислення та ефективніше використання ресурсів (рисунк 1.15), але менше точність (рисунк 1.16).

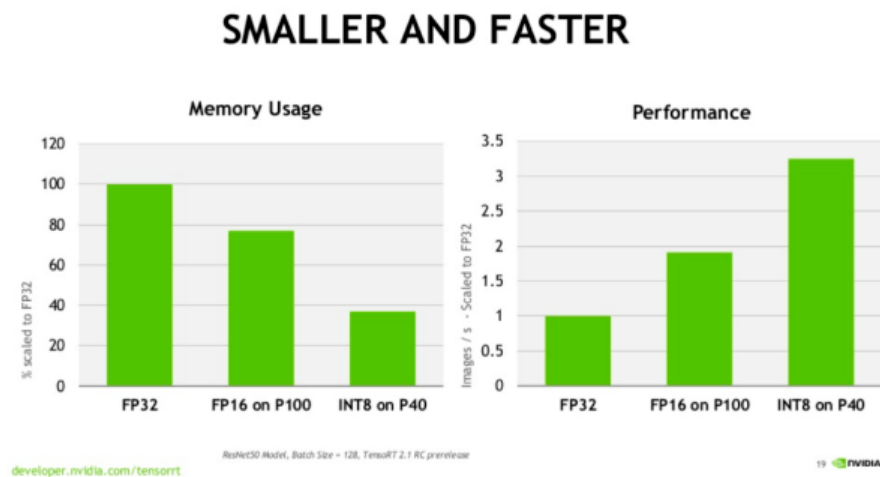


Рисунок 1.15 – Порівняння швидкості FP16, FP32 та INT8

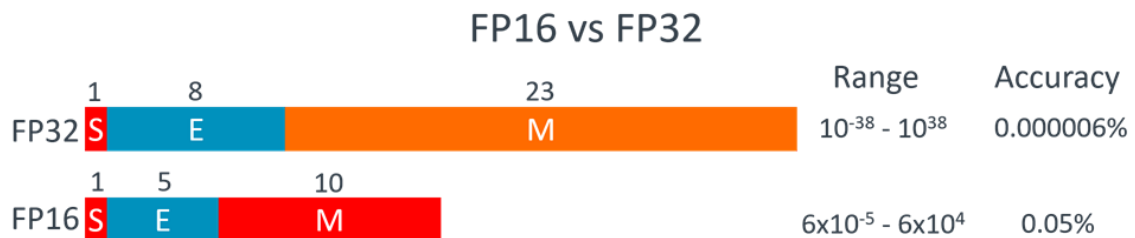


Рисунок 1.16 – Порівняння точності FP16 та FP32

Хоча, як показують дослідження та тренди глибокого навчання, не дивлячись на те, що точність FP32 була популярною досить довгий час, зараз досить багато компаній переходять на FP16, бо додаткова точність майже нічого не дає, але є повільнішою, займає більше пам'яті та знижує швидкість передачі даних між чіпом та оперативною пам'яттю.

Особливу увагу потрібно приділити точності INT8, використовуючи яку моделі нейронних мереж займають мало місця в пам'яті та на диску, а також дають можливість на маленьких форм фактором комп'ютерах по типу Jetson Nano та Google Coral отримувати трильйони операцій за секунду при низькому використанні електроенергії (5-15ват)[6]. Нові моделі GPU (особливо Nvidia) також починають підтримувати точність INT8 (технологія TensorRT), а також розробляють алгоритми квантизації для перетворення вже існуючих моделей з FP16/FP32 в INT8. Порівняння FP32 з INT8 (PTQ та QAT) на серверній відеокарті Nvidia A100 для нейронних мереж та приріст швидкодії зображено на рисунку 1.17. Деякі моделі нейронних мереж демонструють приріст ефективності в 19 раз до/після квантизації.[7][8]

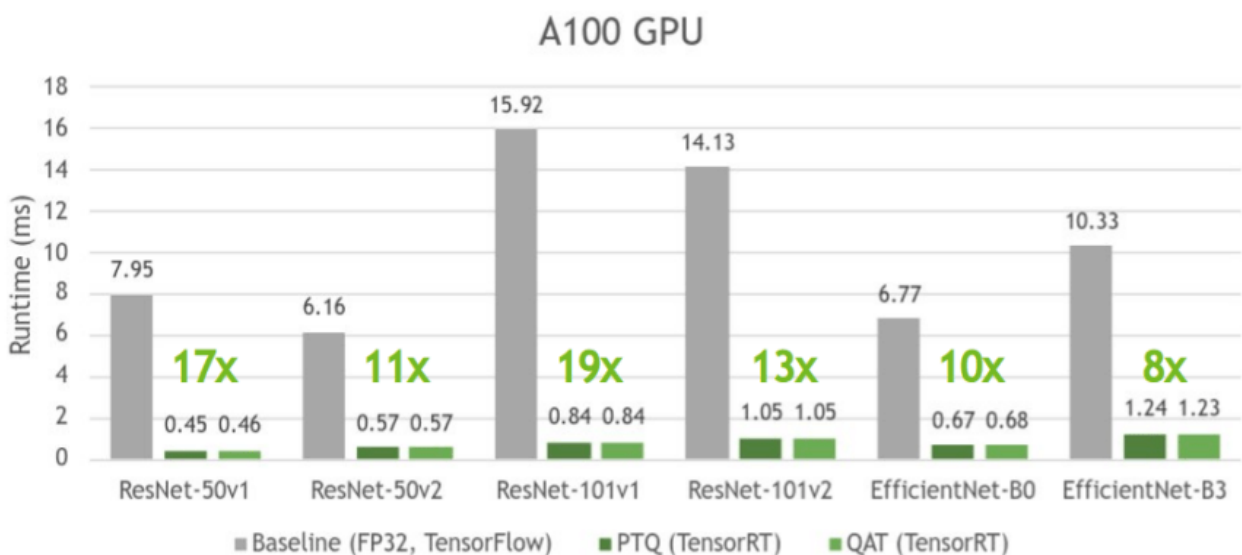


Рисунок 1.17 – Оцінка продуктивності на різних моделях сімейств ResNet та EfficientNet на серверному GPU

Також, оскільки ми дуже часто згадуємо виробника серверних та користувацьких відеокарт Nvidia, потрібно детальніше звернути увагу на описану вище технологію CUDA. Саме ця технологія дає змогу розкрити потенціал оптимізованих під GPU моделей нейронних мереж. Розглянемо основні характеристики:

- CUDA розподіляє обчислювальні завдання на CUDA ядра, що дозволяє використовувати паралельні обчислення;
- CUDA ядра зазвичай розташовані на графічних процесорах і виконують паралельні операції;
- CUDA надає доступ до глобальної та локальної пам'яті на GPU, а також константної та текстурної пам'яті для оптимізації доступу до даних;
- CUDA дозволяє розробникам використовувати кілька GPU разом для вирішення більших обчислювальних завдань.

Деякі бібліотеки повинні бути зібрані з опцією CUDA для підтримки обчислень на графічних ядрах.

1.5 Висновки до розділу 1

У вищезазначеному розділі було розглянуто методи та алгоритми, що використовуються для обробки зображення та виділення на ньому облич та орієнтирів. Розглянуто основні способи підготовки зображення до подальшої обробки, та деякі типи нейронних мереж, що були використані для подальшого аналізу та порівняння методів виокремлення об'єкта на зображеннях.

В результаті аналізу, враховуючи всі плюси та мінуси перерахованих вище алгоритмів розпізнавання обличчя, було обрано реалізацію чотирьох алгоритмів, таких як Каскади Хаара, гістограму орієнтованих графів, підхід глибокого навчання та підхід з використанням холістичних умов. Вибір декількох алгоритмів має на меті збільшити відсоток позитивних пошуків для клієнта.

Для розпізнавання орієнтирів обличчя був обраний підхід на основі моделі Каземі, оскільки це спосіб, який поєднує в собі і швидкість і точність.

2 ПРОЄКТУВАННЯ СИСТЕМИ

В цьому розділі дипломного проєкту буде розглянуто опис основних модулів розробленої системи та доцільність використаних бібліотек, таких як OpenCV, Tensorflow/Caffe, використаних моделей нейронних мереж тощо. Також буде описана взаємодія між головними модулями, детально описана архітектура застосунку/ків та подоланні виклики, які виникли на кожному етапі.

2.1 Загальний опис системи

Як було вже згадано вище, в розробленій системі можна виділити наступні кроки для досягнення поставленої задачі:

- підготовка зображень або відеофайлів та конвертація в один уніфікований формат для подальшого транспортування та обробки;
- можливе транспортування по мережі та/або зберігання в кеші уніфікованих зображень у випадку якщо середовище, на якому запущена система не здатне ефективно обробляти дані;
- отримання та розкодування уніфікованих зображень та/або робота з кешем, перевірка зображень на валідність та можливі оптимізації для прискорення роботи;
- виділення обличчя з розкодованого зображення;
- виділення орієнтирів обличчя з регіонів;
- створення контрольних сум виділених облич у вигляді нормалізованих векторів, які складаються з чисел;
- зберігання векторів та відповідних їм текстових значень (пара ключ - значення);

- порівняння збережених пар між собою для групування облич та відсіювання некоректних результатів;
- відображення результатів.

Виходячи з описаних вище кроків, можна виділити такі основні підсистеми:

- підсистема обробки зображень;
- підсистема кешування зображень;
- підсистема роботи з кешем зображень;
- підсистема виділення облич;
- підсистема виділення орієнтирів;
- підсистема роботи з векторами орієнтирів;
- підсистема відображення результатів;
- підсистема збереження даних.

Результатом об'єднання та роботи цих підсистем в одну загальну є можливість ідентифікації людини на фото та відеофайлах, збереження результатів пошуку для подальшого використання та відображення результатів.

2.2 Вибір архітектури системи

Вибір архітектури системи є критичним кроком при розробці будь-якого проєкту, оскільки від чого залежать ефективність, масштабованість та зручність підтримки системи. Прийняття правильного рішення щодо архітектури передбачає детальний аналіз вимог та обмежень проєкту. Потрібно визначити базові вимоги до функціональності, продуктивності, маштабованості, а також потреби, які можуть виникнути в майбутньому.

Поставленні в пункті 2.1 завдання для реалізації системи накладають свій відбиток на вибір архітектури компонентів системи. Оскільки завдання передбачає використання спеціалізованих графічних та нейронних процесорів

та архітектур на ряду з відображенням результатів ідентифікації користувачу, то вибір архітектури звужується до застосунку для серверу та застунку для клієнта, а також взаємодії між ними. Тобто до моделі розподіленої обчислювальної архітектури, де функціональність програми або додатку розділяється між двома основними типами взаємодіючих компонентів: клієнтами і серверами.

Під сервером будемо розуміти комп'ютер або програмне забезпечення, яке надає певні ресурси, послуги або дані іншим комп'ютерам, які називаються "клієнтами". Сервери зазвичай функціонують постійно, чекаючи на запити від клієнтів.

Під клієнтом будемо розуміти комп'ютер або програму, яка виконує запити до сервера для отримання ресурсів, послуг або даних. Клієнти взаємодіють з сервером, відправляючи запити і отримуючи відповіді.

В реалізованій системі обрана саме така архітектура, скільки в даному випадку ми можемо розділити ролі для серверу і клієнту.

Клієнт (або клієнти) підготовлює зображення для обробки, в тому числі і з відеопотоку та з вже збережених відеофайлів, а також показує результати ідентифікації. Клієнтами можуть бути як веб застосунки, мобільні пристрої з вбудованими камерами, ІОТ пристрої тощо.

Сервер виступає як REST API система, яка буде приймати зображення в визначеному форматі, обробляти та віддавати результат клієнту. Дуже важливими пунктами в цьому випадку є масштабованість, тобто можливість додавати нові ресурси (наприклад, графічні карти) та продуктивність, тобто швидкість роботи.

Розглянемо принцип роботи типового REST API серверу та взаємодію клієнта або клієнтів з ним.

API, або інтерфейс програмування додатків, – це набір правил, які визначають, як програми або пристрої можуть підключатися та спілкуватися один з одним. REST API – це API, який відповідає принципам проєктування REST або архітектурному стилю передачі репрезентативного стану. З цієї

причини REST API іноді називають RESTful API[9] (приклад зображено на рисунку 2.1).

Вперше визначений у 2000 році комп'ютерним науковцем доктором Роем Філдінгом у його докторській дисертації, REST забезпечує відносно високий рівень гнучкості та свободи для розробників. Ця гнучкість є лише однією з причин, чому REST API стали дуже поширеним методом підключення компонентів і програм в архітектурі мікросервісів.

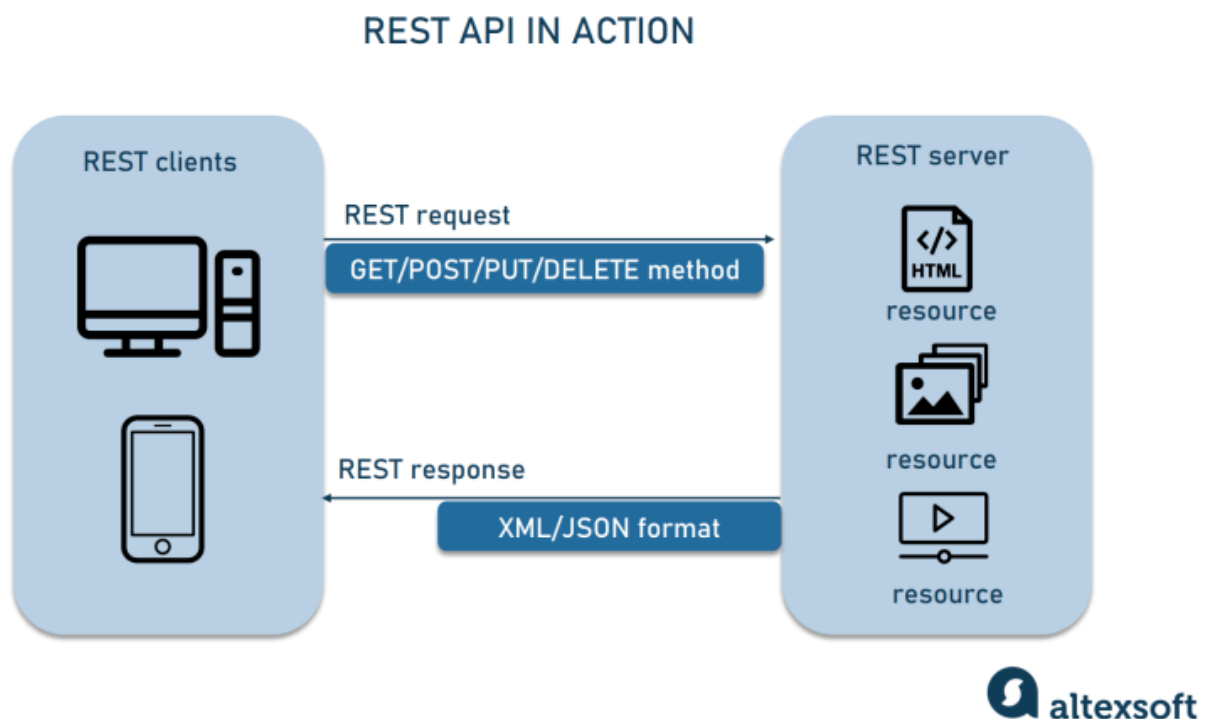


Рисунок 2.1 – Приклад роботи REST API

Розглядаючи RESTful API як інструмент розробки серверної частини застосунку, можна виділити такі принципи:

- уніфікований інтерфейс: усі запити API для одного ресурсу мають виглядати однаково, незалежно від того, звідки надходить запит. REST API має гарантувати, що один і той самий фрагмент даних, як ім'я чи адреса електронної пошти користувача, належить лише одному уніфікованому ідентифікатору ресурсу (URI). Ресурси не повинні бути занадто великими, але повинні містити всю інформацію, яка може знадобитися клієнту;

- розділення клієнту та серверу: у дизайні REST API клієнтські та серверні програми мають бути повністю незалежними одна від одної. Єдиною інформацією, яку має знати клієнтська програма, є URI запитаного ресурсу; він не може взаємодіяти з серверною програмою будь-яким іншим способом. Подібним чином серверна програма не повинна змінювати клієнтську програму, окрім передачі їй запитуваних даних через HTTP;

- без стану: API REST не мають стану, тобто кожен запит повинен містити всю інформацію, необхідну для його обробки. Іншими словами, API REST не потребують сеансів на стороні сервера;

- кешування: якщо це можливо, ресурси повинні кешуватися на стороні клієнта або сервера. Відповіді сервера також повинні містити інформацію про те, чи дозволено кешування для доставленого ресурсу. Мета полягає в тому, щоб підвищити продуктивність на стороні клієнта, одночасно збільшуючи масштабованість на стороні сервера;

- рівнева архітектура системи: в REST API виклики та відповіді проходять через різні рівні. Як правило, клієнтські та серверні програми не підключаються безпосередньо одна до одної. У комунікаційному контурі може бути кілька різних посередників. REST API необхідно розробити так, щоб ні клієнт, ні сервер не могли визначити, чи він спілкується з кінцевою програмою чи посередником.

API REST обмінюються даними через запити HTTP для виконання стандартних функцій бази даних, таких як створення, читання, оновлення та видалення записів (також відомих як CRUD) у межах ресурсу. Наприклад, REST API використовуватиме запит GET для отримання запису, запит POST для його створення, запит PUT для оновлення запису та запит DELETE для його видалення. Усі методи HTTP можна використовувати у викликах API. Добре розроблений REST API схожий на веб-сайт, який працює у веб-браузері з вбудованою функцією HTTP.

В даному розділі ми детально розглянули основні концепції REST API архітектури, дослідили її принципи та розглянули в цілому взаємодію клієнт-сервер

2.3 Розробка структури системи

Розробка структури системи - це важливий етап в процесі створення будь-якої програмної системи або програмного продукту. Виходячи з загального опису системи, та виділеними етапами в розділі 2.1, можна скласти загальну схему роботи системи, що показано на рисунку 2.2.



Рисунок 2.2 – Загальна схема системи

Розглянемо підсистеми, які функціонують окремо в сервері та окремо в клієнті, та опишемо шлях зображення через ці підсистеми.

Під терміном “підсистема обробки зображень” мається на увазі функціонал завантаження даних, які користувач бажає обробити - тобто зображення або відеофайли. Підсистема має здатність завантажувати вказані користувачем файли та обробляти їх незалежно від того, вказано зображення чи відеофайл. Результатом роботи повинен стати масив уніфікованих буферів даних, формат яких розуміє і сервер і клієнт.

Оскільки підсистема дуже тісно зав’язана на обміні даними з користувачем, то вона знаходиться в клієнтському застосунку.

Під терміном “підсистема кешування” або “черга” мається на увазі функціонал накопичення буферів даних задля уникнення ситуацій з мережевими або іншими збоями на клієнті, і на сервері, аби уникнути ситуації, що сервер занадто довго працює з попередніми даними (наприклад, якщо зображення містить багато облич). Підсистема має свою імплементацію в обох частинах системи, хоч вона трохи відрізняється.

Також було введено термін “підсистема виділення облич”, яка по суті своїй робить левову частку всієї роботи. Ця підсистема, використовуючи в своїх підходах нейронні мережі, деякі оптимізації та переваги апаратного середовища, знаходить регіони цікавості, чим полегшує подальші кроки. Вхідними даними цієї підсистеми є уніфіковані буфери даних, а результатом є координати верхньої лівої та нижньої правої точок регіону цікавості, в якому знаходиться один екземпляр обличчя. Дуже важливо, щоб цей крок був максимально швидким, навіть в складних випадках, коли на вхідному зображенні 5-10 облич. Ця підсистема має місце в серверному застосунку.

Подальші кроки з обробки координат регіонів цікавості та перетворення їх в координати орієнтирів обличчя виконує “підсистема виділення орієнтирів”. Вхідними даними цієї підсистеми є координати регіону цікавості та вхідне зображення. Підсистема виділяє з вхідного зображення регіони цікавості, створюючи нові зображення, і на них шукає орієнтири обличчя. Варто зазначити про важливість такого порядку виконання дій - спочатку йде виділення зображень з обличчями, а потім пошук на них орієнтирів. В цьому криється основна оптимізація, яка значно пришвидшує роботу системи в цілому. Оскільки пошук орієнтирів обличчя операція дуже вартісна, зручно зменшити зображення до можливого максимуму (тобто отримати баланс між читабельністю та розміром), паралельно обмежити це зображення так, щоб на ньому було тільки обличчя. Тоді виділення орієнтирів (а це декілька десятків точок) пройде максимально швидко. Ця підсистема також базується в серверному застосунку.

Остання велика підсистема - це “підсистема операцій над векторами орієнтирів”, а саме порівняння, збереження та пошук. Підсистема також відправляє результати пошуку по векторах на клієнтський застосунок, і знаходиться в серверному.

Також варто підкреслити те, що підсистема операцій над векторами орієнтирів займається зберіганням векторів для подальшого завантаження та пошуку по них в майбутньому. Це звичайний масив пар “ключ - значення”, де ключем виступає вектор, а значенням - об’єкт, який зберігає дані про людину.

Підсумувавши розділення підсистем по застосунках, можна прийти до такої більш детальної схеми системи, зображеної на рисунку 2.3.

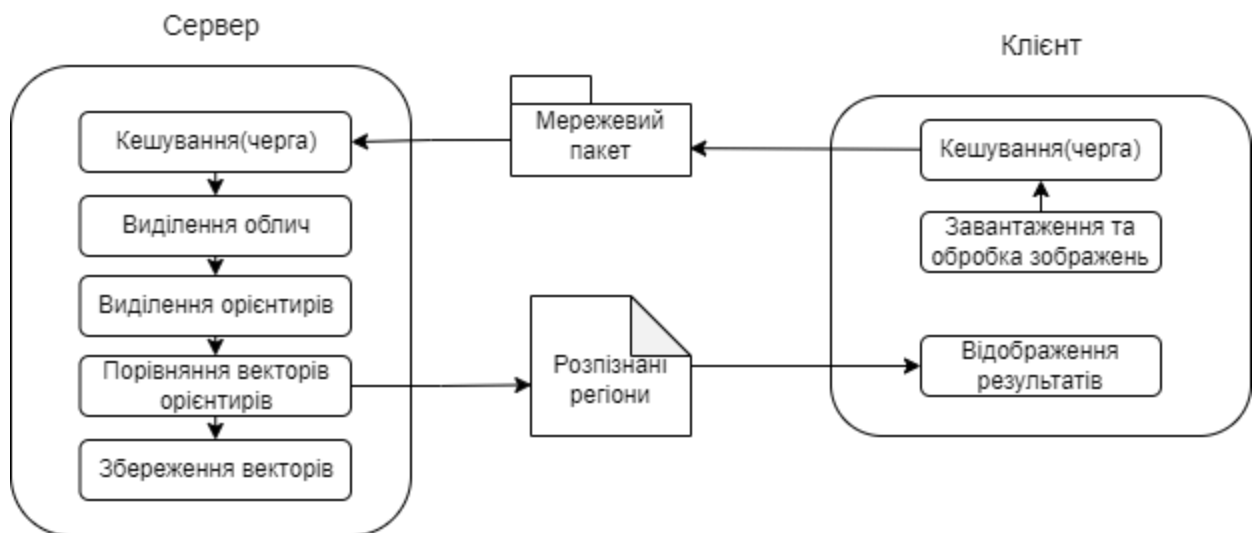


Рисунок 2.3 – Схема підсистем у застосунках

2.4 Функціонування системи

Розглянемо основні алгоритми, бібліотеки та фреймворки, які допомагають системі в цілому та модулям окремо існувати та взаємодіяти ефективно і забезпечувати виконання поставленого завдання.

Насамперед, перед системою стоїть завдання роботи з зображеннями та відео, їх відображення, компресія та декомпресія для пересилки мережею тощо. Також потрібна підтримка роботи в різних середовищах та на різних платформах (як на серверах, що зазвичай передбачає використання операційних систем на основі Linux, так і на клієнтських комп'ютерах, де в перевага у windows), при цьому мати змогу працювати з різними мовами програмування (наприклад, C++ та Python).

Тому лівову частину маніпуляцій з зображеннями, які реалізовані в, наприклад, підсистемах обробки, кешування та відображення зображень забезпечує бібліотека OpenCV (Open Source Computer Vision Library) - бібліотека програмного забезпечення з відкритим кодом та програмного забезпечення для машинного навчання [10]. Основні компоненти бібліотеки схематично зображено на рисунку 2.4)

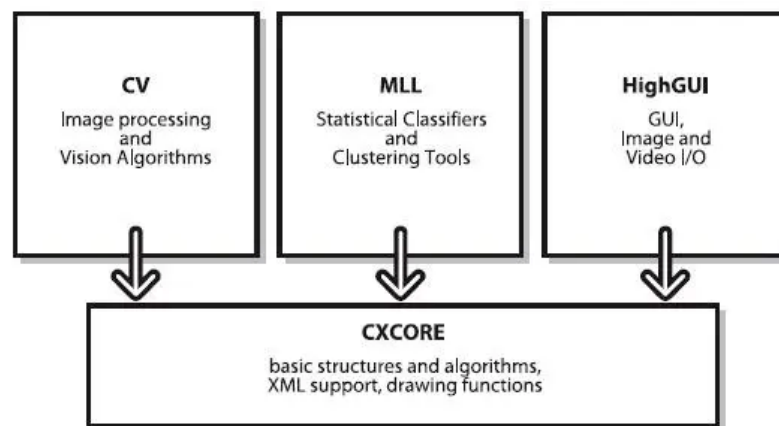


Рисунок 2.4 – Компоненти OpenCV

OpenCV була розроблена спеціально для роботи з комп'ютерним зором та машинним зором, і використовується в багатьох галузях, включаючи розробку програм для розпізнавання облич, аналізу медичних зображень, створення систем безпеки, розробку алгоритмів розпізнавання об'єктів та багато інших застосувань. Бібліотека має понад 2500 оптимізованих алгоритмів, що включає в себе вичерпний набір як класичних, так і сучасних алгоритмів комп'ютерного зору та машинного навчання. Ці алгоритми можна використовувати для виявлення та розпізнавання облич, ідентифікації об'єктів, класифікації дій людини у відео, відстеження рухів камери, відстеження рухомих об'єктів, вилучення 3D-моделей об'єктів, отримання 3D-хмарних точок із стереокамер, скріплення зображень разом для отримання високої роздільної здатності зображення цілої сцени, пошук подібних зображень з баз даних зображень, виправлення червоних очей на зображеннях, зроблених за допомогою спалаху, відслідковування руху очей, розпізнавання краєвидів тощо.

Бібліотека має інтерфейси C++, Python, Java та MATLAB та підтримує Windows, Linux, Android та Mac OS. OpenCV в основному схиляється до додатків зору в режимі реального часу та використовує інструкції MMX та SSE, коли вони доступні. Зараз активно розробляються повнофункціональні інтерфейси CUDA та OpenCL. Існує понад 500 алгоритмів і приблизно в 10 разів більше функцій, які складають або підтримують ці алгоритми. OpenCV написано на мові C++ і має шаблонний інтерфейс, який легко працює з контейнерами STL.

Розглянемо основні можливості та функції OpenCV, які використовуються в системі:

- дозволяє завантажувати зображення та відеопотоки з різних джерел та зберігати їх у уніфікованому форматі;
- є можливість виконувати різні операції зображення, такі як зміна розміру, обрізання, фільтрація, підсвічування та інші маніпуляції з пікселями;
- містить алгоритми та моделі для розпізнавання облич, об'єктів та геометричних форм на зображеннях;

- здатність відстежувати рух об'єктів на відеопотоці та аналізувати його параметри;
- можливість калібрувати камери для точного відтворення розмірів об'єктів, а також робота з тривимірними зображеннями;
- може бути інтегроване з бібліотеками машинного навчання та глибокого навчання для розробки та впровадження моделей і алгоритмів машинного зору.

Тобто ця бібліотека ідеально підходить під завдання, які поставлені перед системою, а саме отримання зображення та підготовку до відправлення на сервер.

Для отримання зображення користувач на застосунку вибирає необхідний файл з носія та натискає кнопку обробки. Бібліотека в свою чергу розбиває на окремі частини, які називаються фреймами та передає ці фрейми далі.

Підготовка до відправки на сервер полягає в наступному. Оскільки сам клієнтський застосунок не займається розпізнаванням, йому потрібно швидко відправити його на сервер для обробки. Бібліотека має вбудовані методи для перетворення з внутрішнього формату в стиснутий по типу jpeg для більш швидкої передачі мереж.

Отриманий після стиснення файл буде переданий по мережі серверу і також розпакований і перетворений у внутрішній формат, використовуючи ту ж саму бібліотеку OpenCV, що підкреслює ще одну перевагу цієї бібліотеки - доступність на різних системах та архітектурах.

Наступним викликом є саме виділення або розпізнавання облич. В цьому випадку в системі реалізовано декілька підходів для виконання поставленої задачі, порівняння ефективності яких на тестових вибірках буде досліджено в наступних розділах. Наведемо вичерпний перелік цих підходів:

- використання каскадів Хаара (Haar Cascades) разом з розглянутих вище бібліотекою OpenCV;
- моделі глибокого навчання (DNN) Caffe разом з бібліотекою OpenCV, що дозволяє використовувати GPU[11];

— моделі глибокого навчання Tensorflow Lite, оптимізовані (квантизовані) для роботи на Edge TPU (процесор від Google, що входить до Google Coral).

TensorFlow - це відкрите програмне забезпечення для чисельних обчислень та машинного навчання, розроблене компанією Google. Воно призначене для побудови та навчання різноманітних моделей глибокого навчання, включаючи нейронні мережі[12]. Із особливостей, TensorFlow використовує модель графу обчислень для визначення та виконання обчислювальних операцій. Граф представляє собою спосіб організації операцій та їх зв'язків.

Також, TensorFlow дозволяє створювати різноманітні моделі глибокого навчання, включаючи згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та інші, використовувати можливості гарячої заміни, векторизації та розпаралелювання для оптимізації обчислень, забезпечуючи швидке виконання складних обчислень, і що саме головне, може використовувати потужності графічних процесорів (GPU) та спеціалізованих пристроїв, таких як Tensor Processing Units (TPU), для прискорення обчислень у машинному навчанні (структура показана на рисунку 2.5).

TensorFlow має активну спільноту користувачів та багато ресурсів, включаючи документацію, уроки та навчальні матеріали, і використовується в різних галузях, таких як комп'ютерне бачення, обробка мовлення, рекомендаційні системи, медичний аналіз та інше.

Варто розглянути різницю між TensorFlow та TensorFlow Lite, оскільки в системі використовується саме Lite версія.

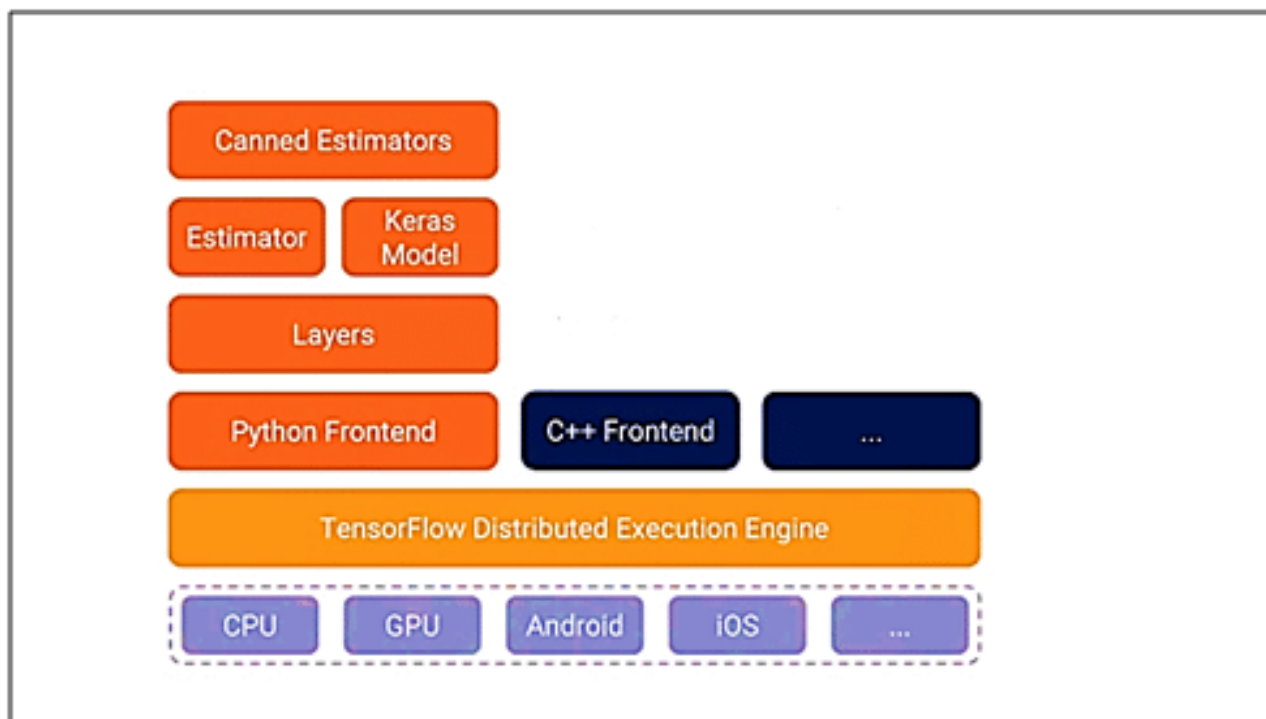


Рисунок 2.5 – Архітектура Tensorflow

TensorFlow Lite був розроблений для мобільних та вбудованих пристроїв, де обмежені обчислювальні ресурси та ефективність є важливими. Він має менший розмір та оптимізований для швидкості виконання на таких пристроях. Одним з таких вбудованих пристроїв являється Google Coral який підтримує квантизовані версії Tensorflow Lite з розміром типу обчислювальних даних INT8, а також оптимізації саме для процесору Edge TPU, апаратного прискорювача для обробки нейронних мереж, оптимізованого для операцій над тензорами та матрицями. На рисунку 2.6 можна побачити порівняльну характеристику CPU (брався 64-bit Intel(R) Xeon@ 3.60GHz), embedded cpu (Quad-core Cortex-A53 @ 1.5GHz, зазвичай це мобільні пристрої) та Google EdgeTPU при використанні різних моделей глибокого навчання.

Також, як в завданні виділення обличчя на зображенні, так і в завданні виділення орієнтирів обличчя можна виділити такі популярні бібліотеки, як Dlib та MediaPipe.

Model architecture	Desktop CPU*	Desktop CPU* + USB Accelerator (USB 3.0) <i>with Edge TPU</i>	Embedded CPU **	Dev Board † <i>with Edge TPU</i>
MobileNet v1	47 ms	2.2 ms	179 ms	2.2 ms
MobileNet v2	45 ms	2.3 ms	150 ms	2.5 ms
Inception v1	92 ms	3.6 ms	406 ms	3.9 ms
Inception v4	792 ms	100 ms	3,463 ms	100 ms

Рисунок 2.6 – Порівняльна таблиця різних типів процесорів

Dlib – це відкрита бібліотека програмного забезпечення на мові програмування C++, яка включає в себе широкий спектр інструментів для розв'язання різних завдань комп'ютерного бачення, включаючи розпізнавання обличч, виявлення об'єктів, відстеження об'єктів, роботу з обробкою зображень та машинним навчанням. Ця бібліотека надає інструменти для виконання різноманітних завдань машинного навчання, включаючи класифікацію, регресію, кластеризацію та інше. Також має потужні засоби для виявлення та розпізнавання обличч, заснованих на гістограмі орієнтованих градієнтів HOG (Histogram of Oriented Gradients)[13] - простому але потужному алгоритмі, основною ідеєю якого є розділення зображення на маленькі області, поєднані між собою, та обчислення гістограми для кожної області. Одним із головних недоліків цього алгоритму є те, що він працює тільки з прямими лицьовими гранями, і погано з повернутими обличчями. Тому цей підхід не використовується як основний в системі. Водночас, dlib пропонує дуже потужний алгоритм пошуку та порівняння орієнтирів на обличчі, який базується на раніше розглянутій моделі Каземі. Модель Каземі навчена на великій кількості зображень з правильними координатами ключових точок обличчя, що дозволяє їй точно визначати ці точки на нових зображеннях.

Імплементація цієї моделі в рамках бібліотеки DLib генерує 68 різних ключових точок (зображено на рисунку 2.7).



Рисунок 2.7 – Орієнтири DLib на обличчі

Наступною бібліотекою є MediaPipe – це відкрита бібліотека та фреймворк, розроблений Google, призначений для розв'язання завдань обробки мультимедіа та розпізнавання об'єктів в реальному часі[14]. Він надає інструменти для роботи з відео- та аудіо потоками, обробки зображень та розпізнавання об'єктів, а також аналізу жестів та взаємодії з реальними об'єктами. MediaPipe створений для розробників, дослідників і інженерів, які працюють у сферах комп'ютерного бачення, аналітики даних та розробки додатків з обробкою відео та аудіо.

MediaPipe також підтримує функціонал розпізнавання обличчя та орієнтирів на ньому, використовуючи раніше описані холістичні моделі - це тип моделі або підходу, який розглядає систему або об'єкт як цілісну сутність, де кожна частина взаємодіє з іншими частинами для створення загального явища або поведінки. У холістичних моделях аналізується взаємодія між всіма складовими частинами системи, а не окремо кожна частина відокремлено. Ця

холістична модель в рамках процесу виявлення ключових точок генерує 468 унікальних точок, які використовуються в побудові меш сітки,приклад якої зображено на рисунку 2.8.

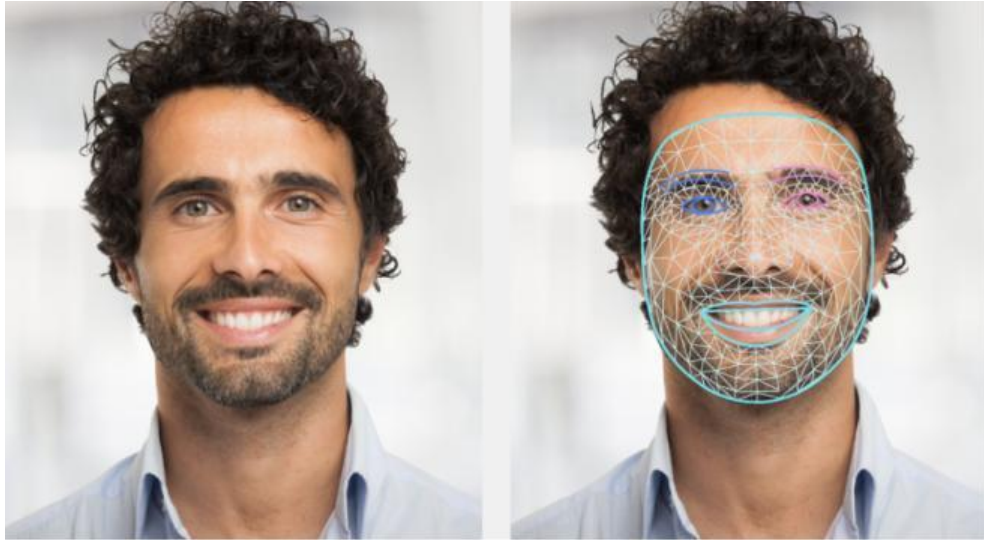


Рисунок 2.8 – Меш-сітка, побудована за допомогою MediaPipe

Якщо порівнювати описані вище бібліотеки в розпізнаванні обличчя та визначенні ключових точок обличчя, то Dlib більш точніший, хоча і генерує менше контрольних точок. Це робить його корисним для завдань, де точність має велике значення, таких як біометрична ідентифікація. Але все ж, у обох бібліотек (як і в задачі розпізнавання облич та контрольних точок в цілому) є задачі, які досі не вирішені, наприклад, різні вирази обличчя і поза голови однієї людини - може значно зменшити точність. Також це умови навколишнього середовища, наприклад, недостатня або занадто сильна освітленість також може сильно вплинути на результат.

Обидві описані вище бібліотеки використовуються в дипломній роботі, як підходи до пошуку лиця, так і для пошуку орієнтирів, хоча в випадку облич перевага надається глибоким нейронним мережам, і тільки у випадку пошуку та порівняння орієнтирів ці бібліотеки стануть у нагоді.

Заключним етапом роботи розробленої системи є збереження сформованої пари ключ значення, де ключ - об'єкт з даними про людину, а

значення - вектор. В системі для цього використовується система управління базами даних PostgreSQL - це потужна об'єктно-реляційна система управління базами даних (СУБД), яка відкрита для використання та розповсюдження. Ця система підтримує SQL (Structured Query Language) і включає в себе велику кількість функцій та можливостей, які дозволяють розробникам зберігати та маніпулювати даними, має вбудовану підтримку транзакцій, що дозволяє виконувати операції з даними з дотриманням атомарності, цілісності, ізоляції та довіри (ACID - atomicity, consistency, isolation, durability). PostgreSQL широко використовується у різних областях, включаючи веб-розробку, програмне забезпечення, аналітику та багато інших сфер, завдяки своїм можливостям, надійності та можливостям розширення. Ця СУБД є вільно розповсюдженим програмним забезпеченням з відкритим вихідним кодом, що означає, що його можете використовувати, змінювати та розповсюджувати без обмежень.

Порівнюючи цю СУБД з іншими подібними, наприклад, MySQL, то можна виділити кілька основних відмінностей в їх функціональності та можливостях. Основною причиною вибору саме PostgreSQL був розширений набір типів даних, таких як географічні, JSON, XML, а також бінарний формат зберігання даних, та більш розвинені можливості управління даними. Оскільки обчисленні нами вектори після серіалізації (один з етапів збереження) мають бінарним формат, то вибір саме цієї СУБД очевидний.

2.5 Висновки до розділу 2

В даному розділі було розглянуто структурну схему реалізованої системи та взаємодію основних підсистем, розділених між клієнтським та серверним застосунками. Щодо серверного застосунку, як основну архітектуру було обрано REST API, і описано, чому ця архітектура найкраще підходить до задач, які виконує система.

Також було розглянуто основні програмні засоби, фреймворки та бібліотеки, такі як OpenCV, Tensorflow та Dlib, які використовуються у реалізованій системі в серверному застосунку, як на етапі розпізнавання обличчя, так і на етапі розпізнавання ключових орієнтирів. Щодо виділення орієнтирів, розглянуто також приклади точок, які генерують бібліотеки.

Розроблено структурну схему всієї системи, та розбито на підсистеми, які в свою чергу розділено між клієнтським та серверним застосунками. Описано основні вимоги до підсистем, їх взаємодію та порядок функціонування.

Заключним етапом розділу було розглянуто приклади систем управління базами даних, таких як PostgreSQL та MySQL, порівняно їх функції та обрано PostgreSQL як СУБД для зберігання ключових точок.

3 РЕАЛІЗАЦІЯ СИСТЕМИ

В цьому розділі дипломного проєкту буде розглянуто опис реалізації основних модулів розробленої системи, тестові вибірки, на яких було протестовані алгоритми розпізнавання облич та алгоритми виділення орієнтирів облич, порівняння їх швидкодії на GPU та NPU, та представлено результати досліджень.

3.1 Визначення навчальних та тестових вибірок

Розділення даних на навчальний набір та на тестовий набір даних є одним з ключових понять у машинному навчанні та аналізі даних. Вони використовуються для навчання моделі та оцінки її точності відповідно.

Навчальна вибірка - це частина набору даних, яка використовується для тренування моделі. Дані з навчальної вибірки використовуються для того, щоб модель "навчилася" на них, тобто зрозуміла закономірності та зв'язки між ознаками та вихідними даними. Оскільки для пошуку облич ми використовуємо вже навчені та сотні тисяч зображень нейронні мережі в складі пакетів бібліотек або додатково завантажені з мережі, то зосередимося лише на описі тих моделей, які використовуємо, тобто на моделях Каскадах Хаара, НОГ детекторі та CNN моделях бібліотеки Dlib, окремо завантаженої мережі глибоко навчання Res10, моделях бібліотеки MediaPipe та моделі для Edge TPU.

Розмір навчальної вибірки для каскадів Хаара може значно варіюватись залежно від типу та складності об'єкта, який потрібно виявити, а також від вимог до точності та робочих умов. Однак, загалом, чим більше навчальних даних, тим краще для навчання моделі. Для того щоб отримати більш-менш ефективний каскад, достатньо навчити його на вибірці розміром від декількох сотень до тисячі зображень. В нашій системі використовується *haarcascade_frontalface_default.xml*, вже навчений на декількох тисячах позитивних зразків (зображень), які містять цільовий об'єкт (наприклад, в нашому випадку, це обличчя), і на достатній кількості негативних зразків, на яких об'єкт відсутній.

Щодо бібліотеки DLib, вона, як було зазначено вище, містить підхід глибокого навчання CNN, який базується на моделі *mmod_human_face_detector.dat*, яка має складну структуру, підтримує GPU та натренована на навчальній вибірці в десятки тисяч зображень, зібраних з різних ресурсів, високоточний і дуже надійний, здатний розпізнавати обличчя під різними кутами огляду, умовами освітлення та оклюзією, а також НОГ + Linear SVM підхід, який покладається на обчислення в реальному часі, але дуже чутливий до змін кута огляду на зображення.

Також варто розглянути майже стандартну для пакету OpenCV модель глибокого навчання *res10_300x300_ssd_iter_140000.caffee* - це назва моделі об'єктного виявлення, яка використовується у бібліотеці OpenCV для виявлення

облич на зображеннях за допомогою алгоритму Single Shot Multibox Detector (SSD).

- "res10" вказує на те, що це ResNet (Residual Network) з меншою кількістю шарів, наприклад, ResNet-10 або інша версія з 10 шарами;
- "300x300" означає розмір зображення, який використовується для обробки моделлю. У цьому випадку модель працює з зображеннями розміром 300x300 пікселів;
- "ssd" вказує на те, що використовується алгоритм виявлення об'єктів Single Shot Multibox Detector, який дозволяє виявляти обличчя та інші об'єкти на зображенні одним шотом;
- "iter_140000" позначає кількість ітерацій (у цьому випадку 140000), тобто кількість разів, які модель була навчена на навчальному наборі даних під час тренування.

В свою чергу, термін ResNet-10 в комбінації з SSD є деяким варіантом нейронної мережі ResNet (Residual Neural Network) з 10 вершинами, але з меншою кількістю шарів у порівнянні з більшими версіями ResNet, але вона все ще здатна виконувати досить складні завдання класифікації та виявлення об'єктів. Комбінація цієї мережі з SSD дозволяє відносно точно та швидко виявляти об'єкти на зображеннях. Ця модель є популярним вибором для виявлення облич у реальному часі через її швидкодію та здатність до точного визначення облич на зображеннях.

Бібліотека MediaPipe використовує моделі типу MobileNet-v2 CNN[15], що в свою чергу означає модель глибокого навчання для мобільних архітектур зі скоригованими блоками для досягнення розпізнавання в реальному часі, яка використовує точність FP16, і може використовуватись фреймворком TensorFlow Lite.

І, нарешті, бібліотека PyCoral, яка забезпечує ефективну роботу Edge TPU, за аналогією з MediaPipe використовує MobileNet-v2 для TensorFlow Lite, але її скомпільовану квантизовану версію для Edge TPU.

Для тестування обраних підходів та моделей глибокого навчання було обрано декілька цільових зображень з 1 лицем (зображено на рисунку 3.1), які повинні показати найшвидші результати нейронних мереж, з 5 обличчями (зображено на рисунку 3.2) - середній результат та можливі дефекти розпізнавання та 10 обличчями (зображено на рисунку 3.3) - повинні показати, як нейронні мережі справляються з великою кількістю людей. Ідея в тому, щоб дослідити швидкодію та ефективність мереж на дуже малих, середніх та дуже великих об'ємах розпізнавання об'єктів, включаючи різні раси, статі та положення облич.



Рисунок 3.1 – Одна жінка

Для наочності, продемонструємо окремо результати розпізнавання окремо на Coral TPU, та на GPU (використовується GeForce RTX 3050 Mobile), оскільки деякі моделі не працюють на обох тестових майданчиках, наприклад, ResNet 10 SSD не вдалося протестувати, бо потрібно більше ресурсів, ніж є на мобільному пристрої.



Рисунок 3.2 – Група з 5 людей різних статей, рас та положень обличчя



Рисунок 3.3 – Група з 10 облич людей різних статей та рас

Спершу розглянемо швидкодію мереж на GPU. На рисунку 3.4 зображено порівняння чотирьох описаних раніше підходів до виявлення одного обличчя, середній, мінімальний та максимальний час виявлення в мілісекундах на протязі 100 циклів, виявленні і виділені зеленим регіони з обличчям, а також побудований графік затраченого часу на протязі всіх 100 циклів розпізнавання. На рисунках 3.5 та 3.6 відповідно зображені всі дані для п'яти та десяти облич.

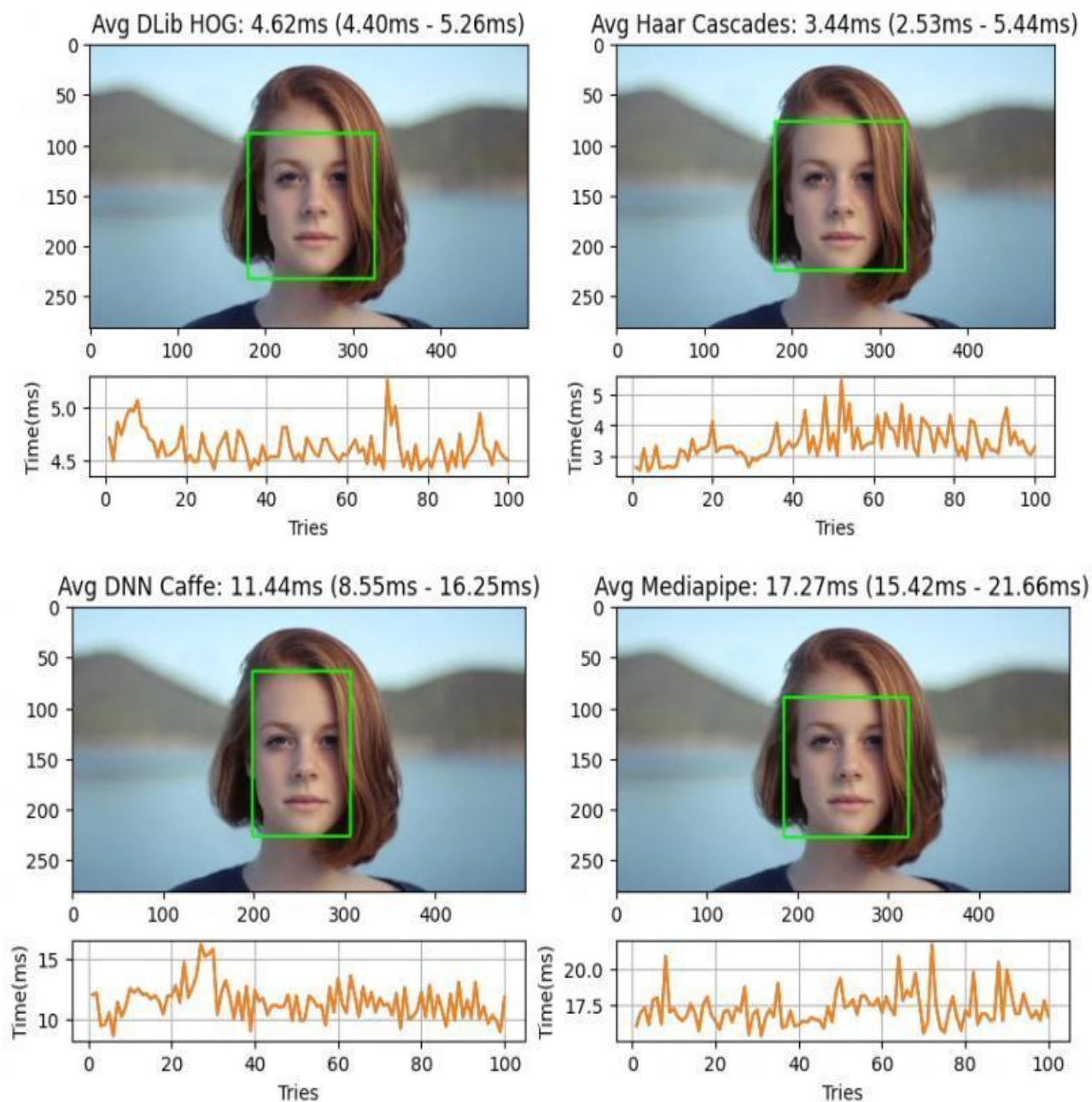


Рисунок 3.4 – Результат розпізнавання одного обличчя на GPU

Якщо коротко охарактеризувати представлені вище результати, то всі алгоритми та моделі показують непоганий результат за швидкістю роботи і показують мінімальну зміну в результатах при збільшенні числа об'єктів. Окрему потрібно виділити DNN Caffe, який показує хороші результати на 1 обличчі і дуже хороші на великій кількості, в той час як MediaPipe пасе задніх майже всюди, а в останньому випадку і взагалі розпізнав 8 з 10.

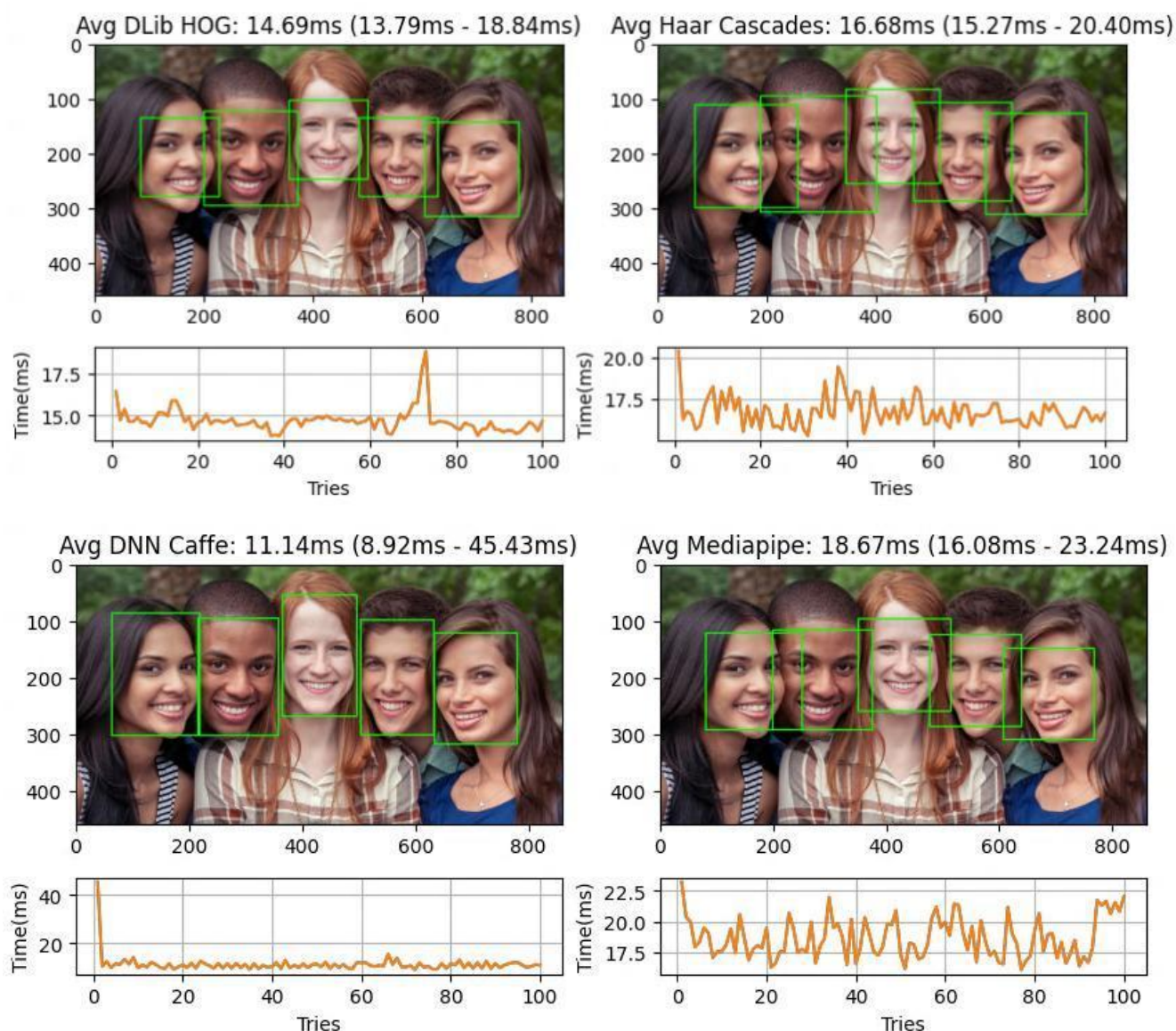


Рисунок 3.5 – Результат розпізнавання п'яти облич на GPU

Також доволі цікаво, чому за 100 циклів формується така велика різниця між мінімальним та максимальним часом інтерференції, іноді навіть більше 10 мілісекунд. Скоріш за все, оскільки відеокарта не серверна, то іноді ресурси потрібні і для обробки монітору.

Далі розглянемо наочно, як працює виділення орієнтирів зображення. Для цього використаємо підходи бібліотек DLib та MediaPipe та виміряємо швидкодію на зображеннях з одним, п'ятьма та десятьма обличчями [16][17]. На рисунках 3.7, 3.8 та 3.9 відповідно показані результати виділення орієнтирів для цих зображень.

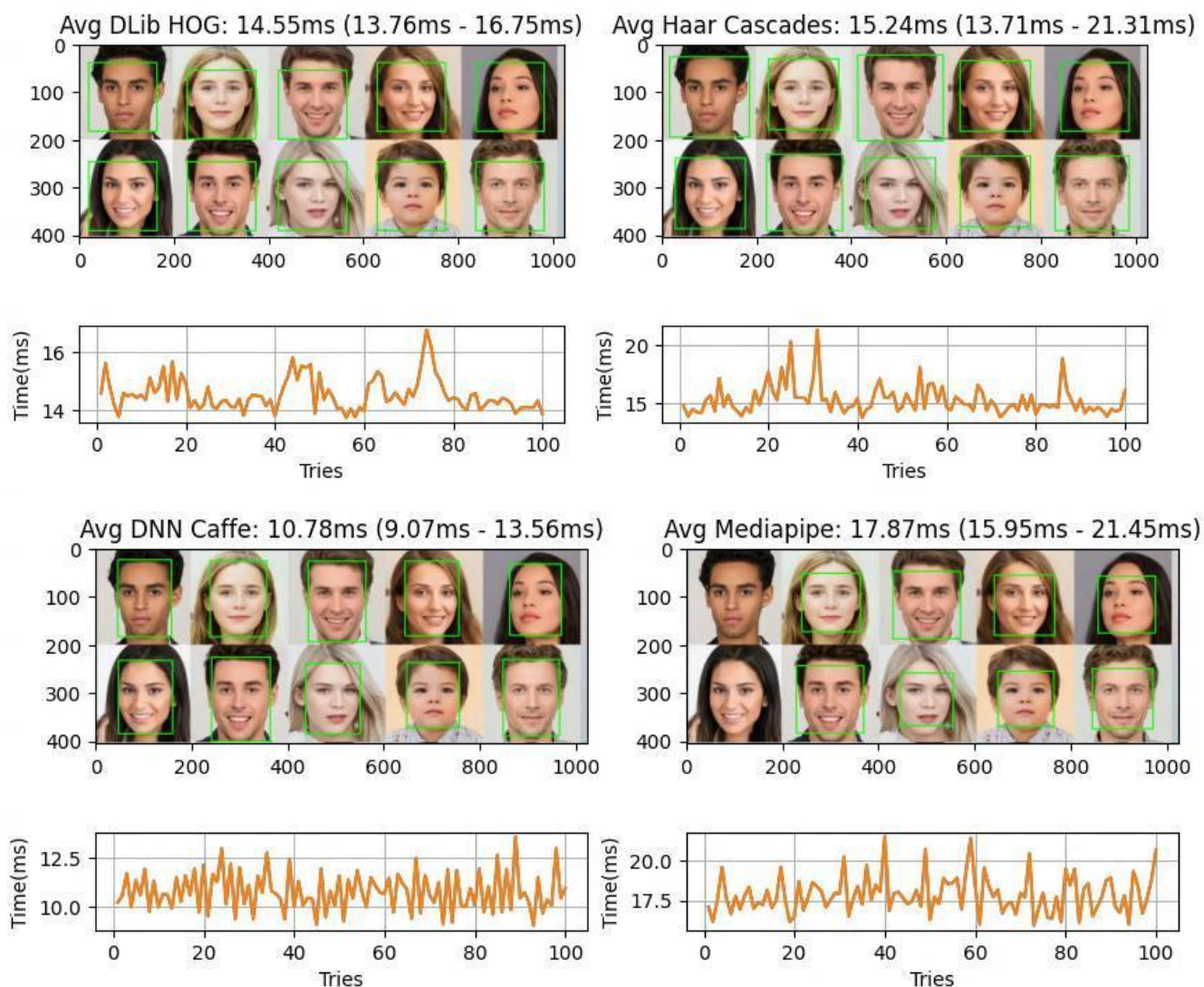


Рисунок 3.6 – Результат розпізнавання десяти облич на GPU

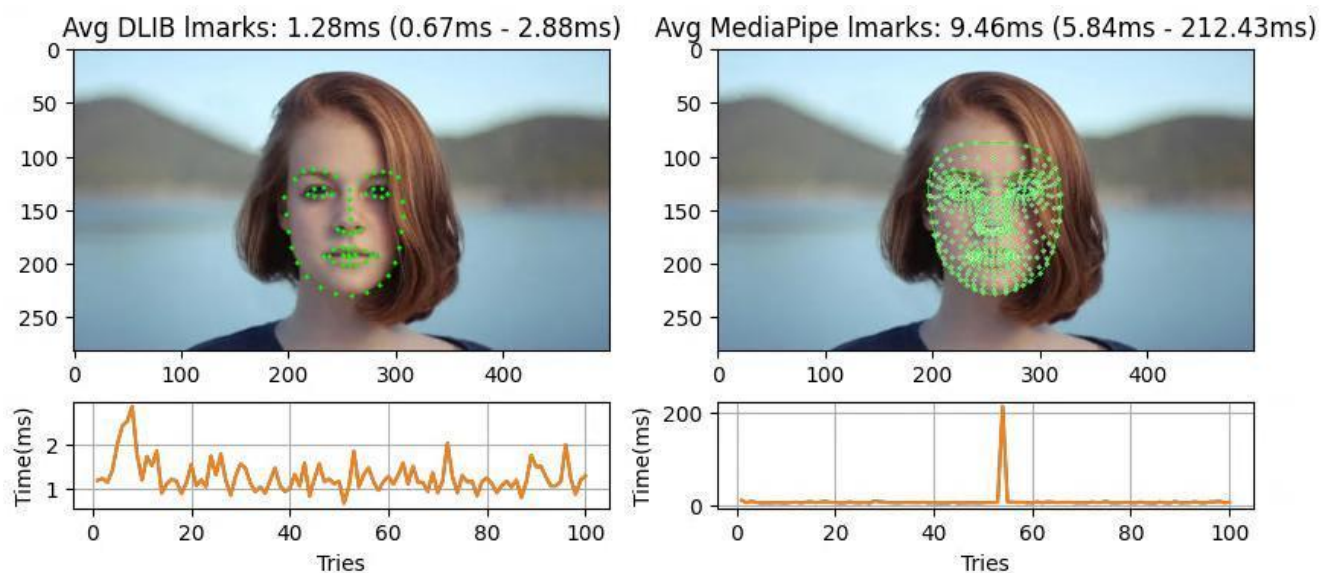


Рисунок 3.7 – Результати виділення орієнтирів для обох бібліотек на 1 обличчі

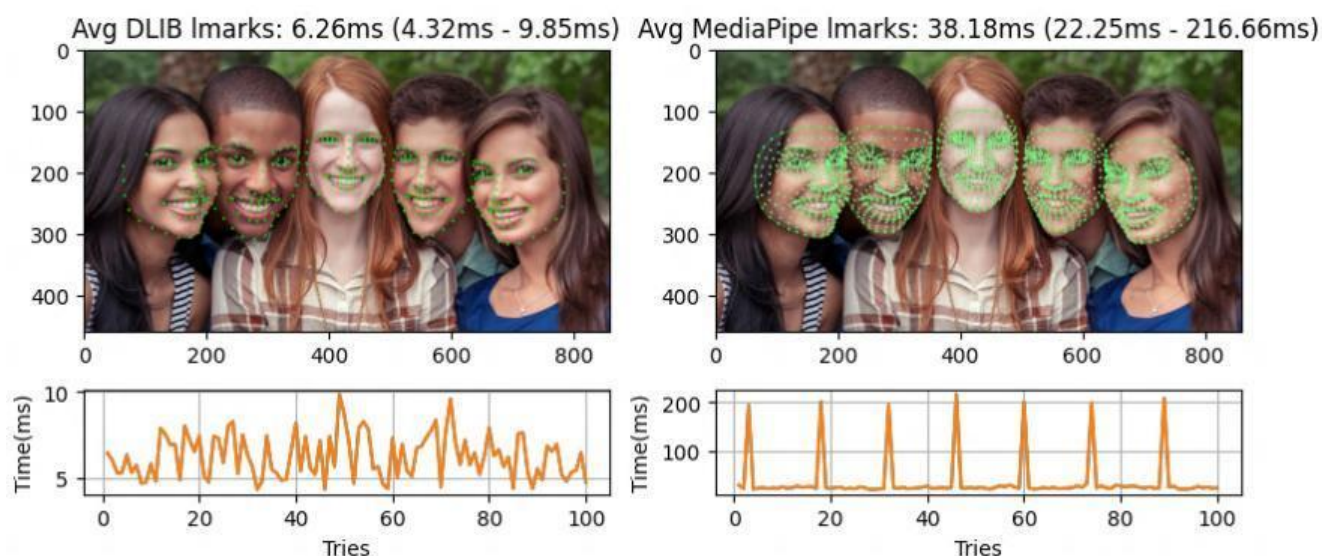


Рисунок 3.8 – Результати виділення орієнтирів для обох бібліотек на 5 обличчях

Розглянувши результати тестів пошуку орієнтирів на обличчі, можна дійти висновку, що обидві бібліотеки працюють непогано, але MediaPipe все ж гірше. Це зумовлено тим, що в апі бібліотеки DLib закладена можливість пошуку у вже знайдених раніше регіонах, наприклад, на вже знайдених іншими, більш швидкими алгоритмами, регіонах з обличчями. MediaPipe, в свою чергу, попередньо шукає обличчя (знову) та орієнтири, що зменшує швидкодню в декілька раз. Також MediaPipe шукає 468 орієнтирів, на відміну від 68 у DLib, що також додає свій час до швидкодії.

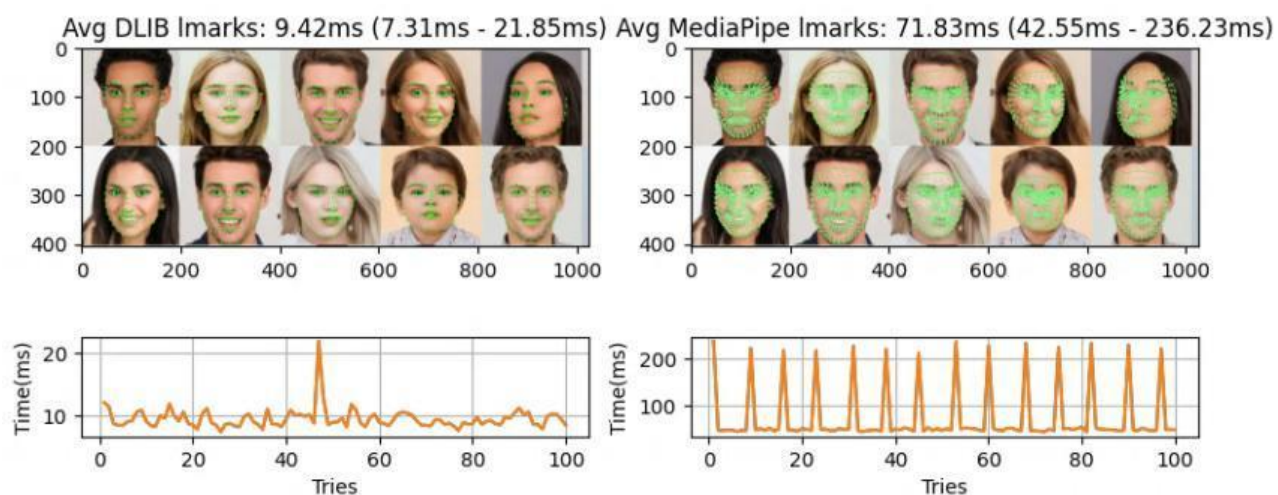


Рисунок 3.9 – Результати виділення орієнтирів для обох бібліотек на 10 обличчях

Найголовнішим мінусом Mediapipe є незрозумілі стрибки часу роботи майже в 5 раз - з 40 до 200+ мілісекунд, що може дуже сильно погіршити роботу всієї системи.

3.2 Реалізація модулів системи

Розглянемо детально модулі окремо підсистеми сервер та підсистеми клієнт. Сервер системи ідентифікації людини реалізований на мові Python 3, та містить в собі імплементацію REST API, набір пов'язаних класів та схему СУБД.

REST API містить чотири ендпоінти, відповідальні за прийом повідомлень з клієнту в форматі JSON, та схематично зображений на рисунку 3.10 (формат стандарту OpenAPI).

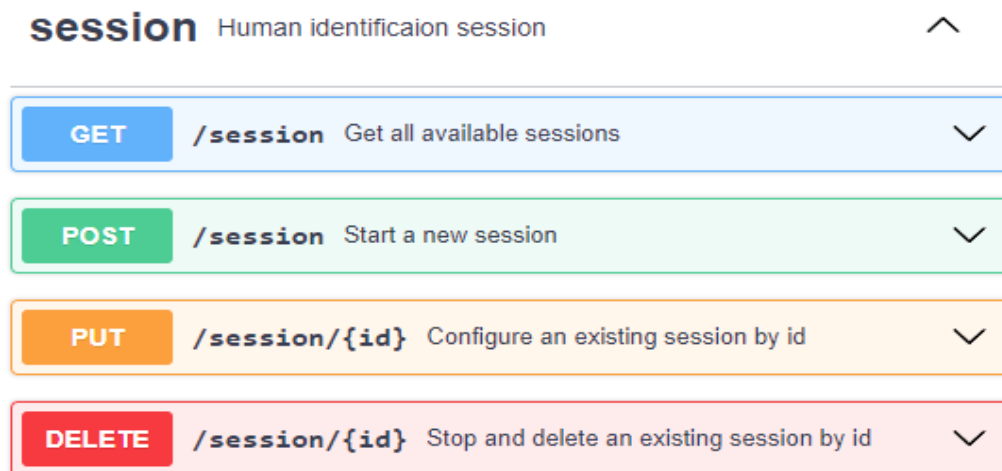


Рисунок 3.10 – REST API серверу у форматі OpenAPI

Ендпоінт GET /session використовується для отримання списку всіх сесій та повертає масив ідентифікаторів активних сесій в форматі json, наприклад:

```
{[
  "69b751c1dcceaef43beb5342f372a49b",
  "7cb2122c823913c9ff870c48e6cd5ffc"]}
```

Ендпоїнт POST /session використовується для створення нової сесії, яка є уособленням того, як сервер бачить клієнта. Саме при створенні сесії генерується унікальний ідентифікатор клієнта, створюються всі необхідні об'єкти на сервері для роботи з зображеннями та знайденими точками тощо. Після необхідних викликів ендпоїнт повертає наступне повідомлення в форматі json, яке містить хеш сесії або ідентифікатор сесії, порт підключення сокету для передачі бінарних даних та статус виконання команди:

```
{
    "cmd": "start_session",
    "status": "success",
    "session": "69b751c1dcceaef43beb5342f372a49b",
    "port": 50778
}
```

Ендпоїнт PUT /session використовується для конфігурування сесії за вказаним ідентифікатором. Клієнт, використовуючи даний ендпоїнт, говорить серверу які операції потрібно виконати над зображенням, переданим раніше через сокет, відкритий при створенні сесії. Операції, які сервер підтримує:

— *FaceRecognition* - пошук регіонів, що можуть містити обличчя. Також ця операція потребує назви алгоритму для пошуку обличчя як додаткового параметру. Повертає знайдені регіони через сокет у вигляді бінарних даних;

— *FaceLandmarks* - пошук орієнтирів на обраному регіоні обличчя. Повертає знайдені масиви орієнтирів через сокет у вигляді бінарних даних;

— *SaveFace* - збереження вирахованих орієнтирів обраного регіону до бази даних. Потребує ідентифікатору знайденої людини як додаткового параметру;

— *SearchFace* - пошук обраного на фото обличчя по базі даних. Повертає ідентифікатор людини та відсоток співпадіння через сокет у вигляді бінарних даних.

Ендпоїнт DELETE /session видаляє сесію з масиву та закриває сокет. Повертає результат виконаної операції.

Описані вище ендпоїнти керують станом серверу та створюють об'єкти сесій, і ті в свою чергу створюють об'єкти інших класів. Розглянемо взаємодію та будову цих класів (див.рис.3.11).

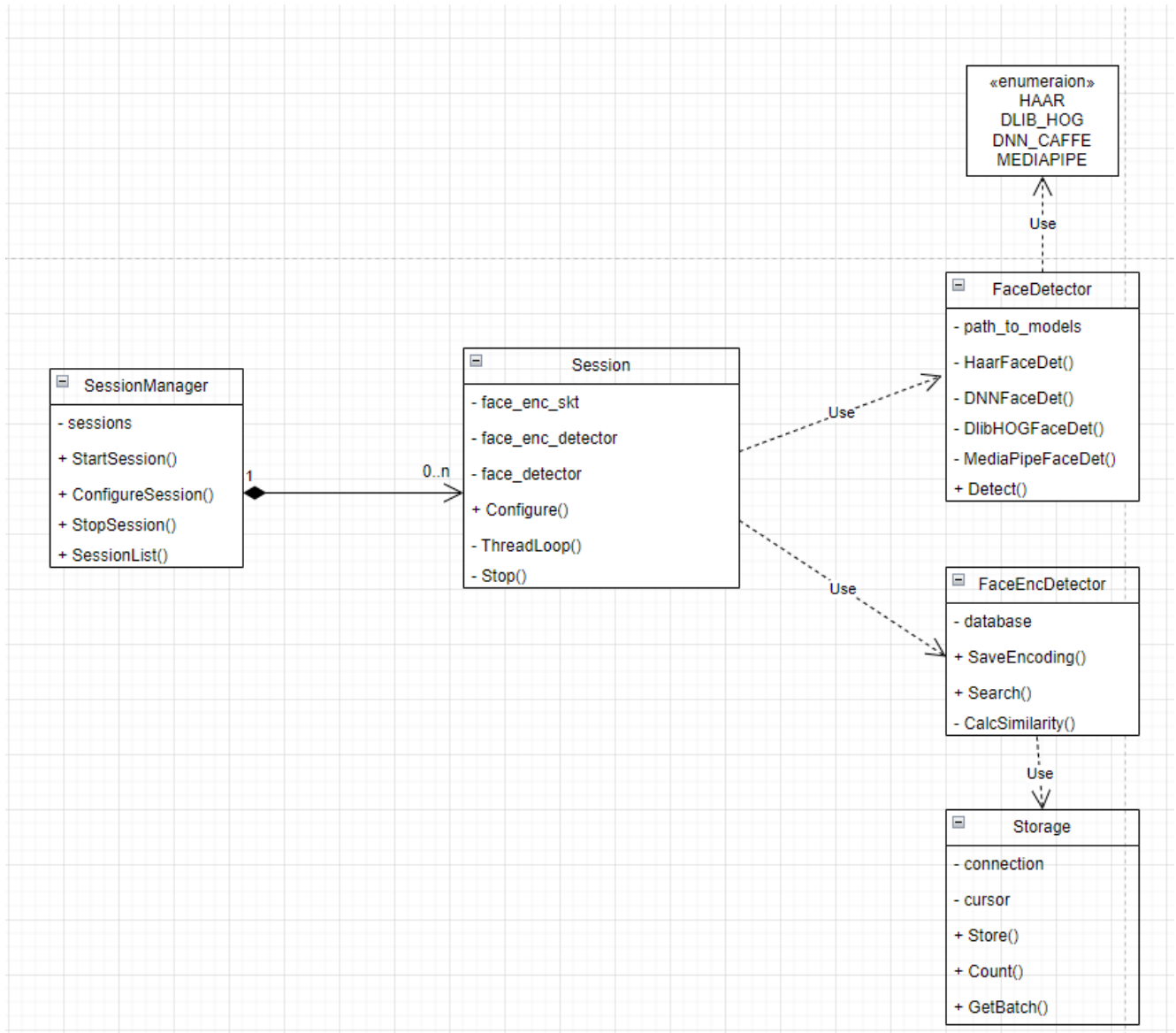


Рисунок 3.11 – Діаграма класів серверу

На рисунку 3.11 зображено діаграму класів серверу:

- клас *SessionManager* відповідає за створення, конфігурування та видалення сесії;

- клас *Session* відповідає за операції з сокетом, по якому приходять бінарні дані та за вибір операцій, пов'язаних з розпізнаванням, в залежності від поточної конфігурації;
- клас *FaceDetector* відповідає за вибір алгоритма пошуку обличчя в залежності від конфігурації сесії;
- клас *FaceEncDetector* відповідає за збереження знайдених орієнтирів до бази даних, пошук співпадінь по заданому критерію та для підрахунку схожості між заданим критерієм та збереженими даними;
- клас *Storage* відповідає за операції з базою даних PostgreSQL, а саме вставка, вибірка, кількість строчок.

Оскільки всі дані зберігаються в базі даних PostgreSQL, варто розглянути її структуру та наявні таблиці. На рисунку 3.12 зображена структура єдиної таблиці *encodings*, яка містить в собі всі збережені орієнтири у вигляді ключ-значення (вигляд таблиці показано на рисунку 3.12):

- поле *label* типу *text* зберігає в текстовому вигляді інформацію про людину, наприклад ім'я;
- поле *encoding* типу *bytea* зберігає в бінарному вигляді інформацію про збережені серіалізовані орієнтири обличчя.

	label text	encoding bytea
1	single_girl	[binary data]
2	boy4	[binary data]

Рисунок 3.12 – Таблиця *encodings* з даними

Після того як було описано структуру класів серверного застосунку та взаємодію між ними, то можна навести діаграму послідовності, яка показує послідовність взаємодій між об'єктами або компонентами в серверній системі. Ця діаграма (зображено на рисунку 3.13) відображає послідовність повідомлень, які передаються між об'єктами або компонентами у певній сценарії взаємодії.

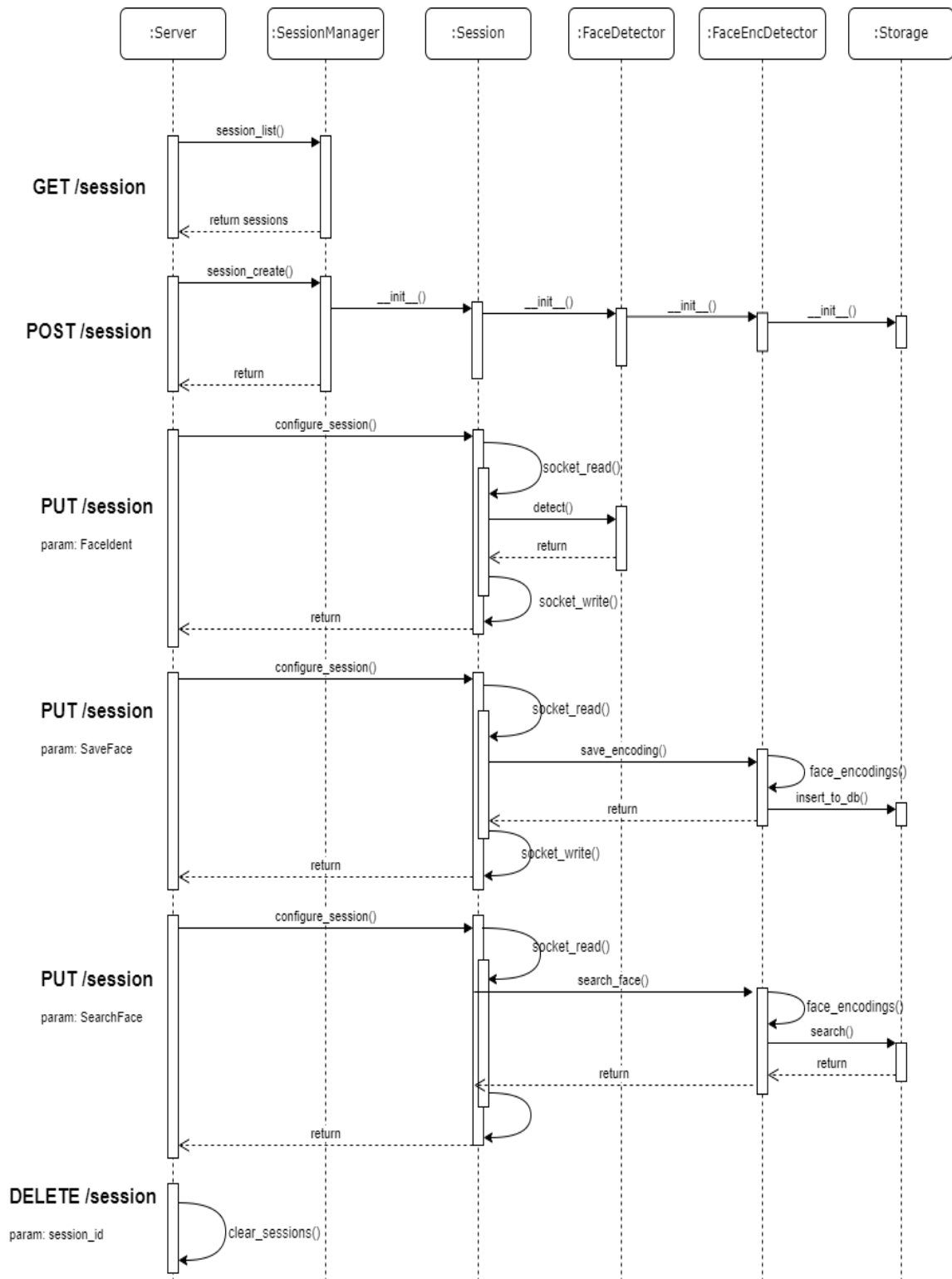


Рисунок 3.13 – Діаграма послідовностей серверного застосунку

Клієнт системи ідентифікації людини представляє собою десктопний застосунок реалізований на мові C++ за допомогою фреймворку Qt6. Застосунок має мінімалістичне керування, проте на 100% відсотків виконує поставлену задачу: завантаження та відображення ресурсів (зображень та

відео), перемотка відео, обмін даними з сервером, пошук та відображення облич, пошук та збереження вибраного обличчя. На рисунку 3.14 зображений інтерфейс клієнтського застосунку з вибраним ресурсом у вигляді зображення зі знайденим одним обличчям жінки, вибраним для пошуку (зелений квадрат), та результатом пошуку в нижній частині вікна.

Структура клієнтського застосунку проста і складається всього з одного класу `MainWindow`, який реалізує всі елементи інтерфейсу, відображення фото або відео у спеціально виділеній області, обробку запитів на сервер та відповідей від нього, що включає в себе асинхронні запити до серверу, парсинг відповідей у форматі `json`, відкриття та закриття сокету для ефективної передачі зображень тощо.

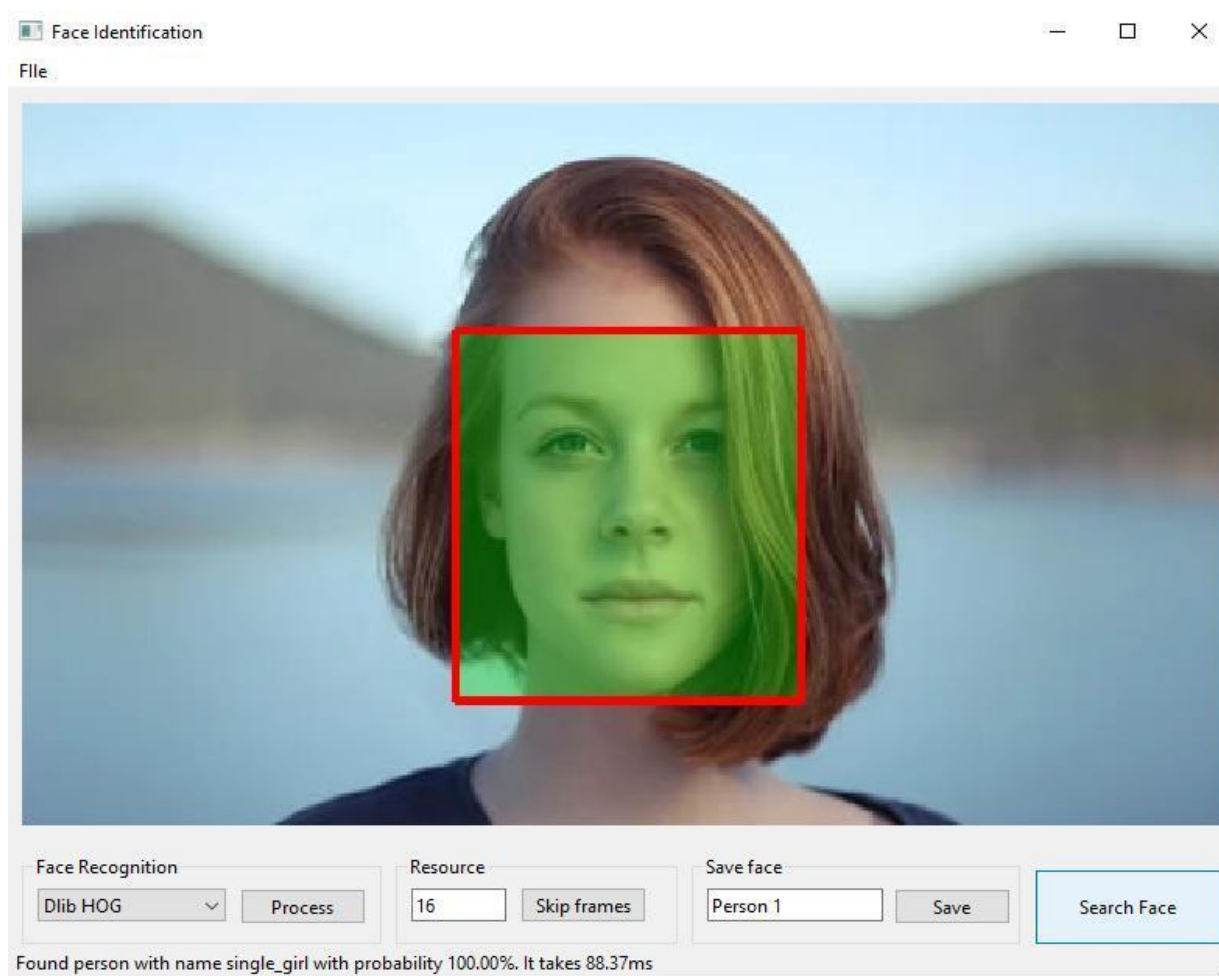


Рисунок 3.14 – Клієнтський застосунок

3.3 Висновки до розділу 3

У вищезазначеному розділі було розглянуто практичне застосування описаних в розділах раніше та використаних в системі бібліотек, таких як DLib, OpenCV, Tensorflow та Mediapipe та короткий огляд моделей глибокого навчання, використаних разом з цими бібліотеками.

Для кожного з підходів розпізнавання обличчя було проведено порівняння швидкості роботи в різних умовах роботи, а саме коли на зображенні одне, п'ять та десять облич, а також побудовано графік для кожного підходу і кожної умови та на протязі ста ітерацій. Це показало ефективність та стабільність кожного з алгоритмів, що в свою чергу вплинуло на доцільність використання тих чи інших підходів до вирішення проблем пошуку облич. Так, наприклад, завдяки цим дослідженням було виявлено цікаві особливості роботи бібліотеки Mediapipe як на етапі пошуку обличчя, так і при пошуку орієнтирів, а саме незрозуміле збільшення часу обробки зображення в кілька разів, що є неприпустимим.

Було описано структуру реалізованих серверного та клієнтського застосунків, розроблено та описано структуру класів, розроблених в рамках дипломного проекту, наведено діаграму послідовностей для серверного застосунку та приведено приклад роботи клієнтського застосунку.

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Для того, щоб відобразити результати роботи проекту з мережі Інтернет було взято декілька фото відомих людей в якості навчальної вибірки, тобто є контрольна інформація, хто ця людина, і під яким тегом вона потрапить в базу даних. Також було взято відеофайли та фото, на яких ця людина присутня, але в дещо зміненому вигляді, які будуть виступати як тестова вибірка.

У роботі визначається, на скільки добре система ідентифікує ту чи іншу людину та скільки часу буде витрачено на виконання цього.

4.1 Тестування системи

В рамках тестової задачі було взято фото високої якості у вільному доступі відомого американського співака Честера Беннінгтона, відомого своїми творами в групі Linkin Park, та відео їх синглу Numb (2.1 мільярди переглядів на платформі Youtube).

Після відкриття клієнтського застосунку, користувач, впевнившись в тому, що підключення до серверу успішне (в статус-барі повинно бути відповідне повідомлення про з'єднання з сервером), завантажує фото з навчальної вибірки до застосунку. Як клієнтський застосунок виглядає після підключення до серверу та завантаження фото, можна подивитись на рисунку 4.1.

Оскільки клієнтський застосунок працює з навчальною вибіркою, тобто всі подальші тестові операції будуть засновані на цій навчальній вибірці, вибране фото повинно бути максимально чітким та щоб на нього було видно всі деталі обличчя, наприклад очі, ніс та губи.

Після завантаження фото, користувач має виконати крок пошуку обличчя на фото, вибравши один з декількох запропонованих алгоритмів. Оскільки ці алгоритми працюють по різному, то можуть давати і різні регіони цікавості, тому користувач має змогу вибрати найбільш підходящий йому регіон, у який входить максимум площі обличчя. Після цього користувач натискає кнопку *Process* та перевіряє результат пошуку обличчя.

В даному випадку каскади Хаара дали непоганий результат при 10 мілісекундах роботи. Результат можна подивитись на рисунку 4.2.

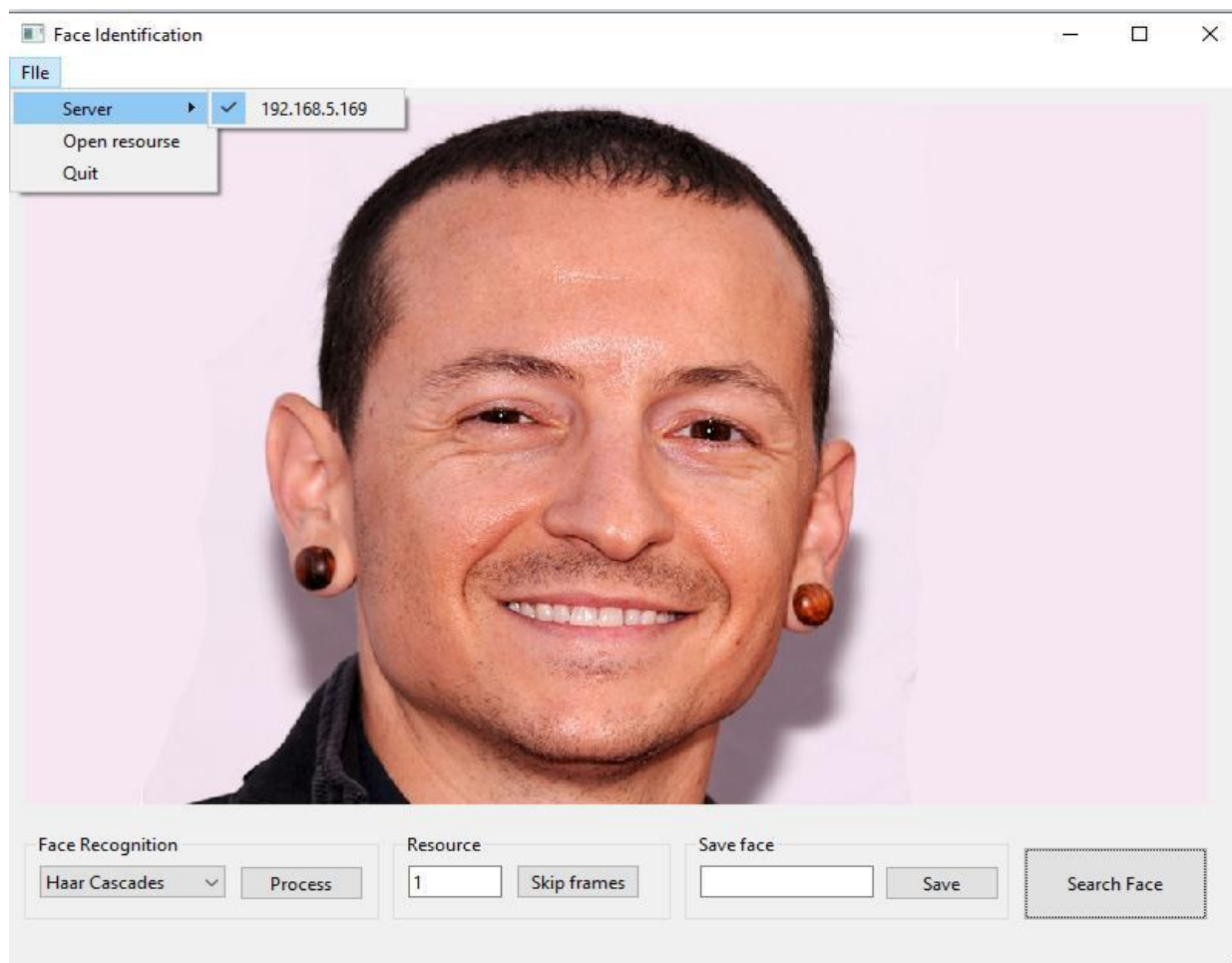


Рисунок 4.1 – Фото Честера як навчальна вибірка для задачі 1

Якщо користувач задоволений результатом, то він має вибрати регіон (клікнувши всередину червоного квадрата, вписати тег людини в поле *Save face* та натиснути кнопку *Save*.

Клієнтський застосунок в цьому випадку відправляє вибрану користувачем область на сервер, де той в свою чергу, застосовуючи наявні алгоритми, вираховує орієнтири обличчя, серіалізує їх, зберігає в базу даних під вказаним тегом та відправляє результат операції на клієнтський застосунок. Приклад успішного зберігання навчальної вибірки можна побачити на рисунку 4.3.

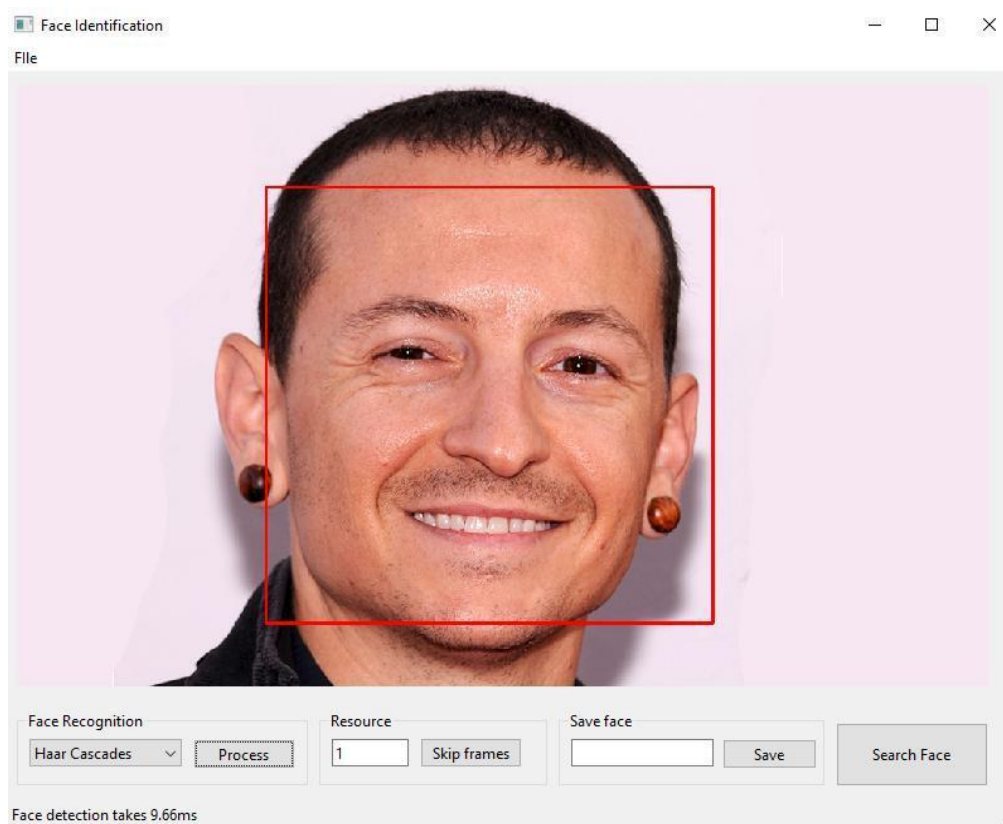


Рисунок 4.2 – Результат першого етапу роботи застосунку

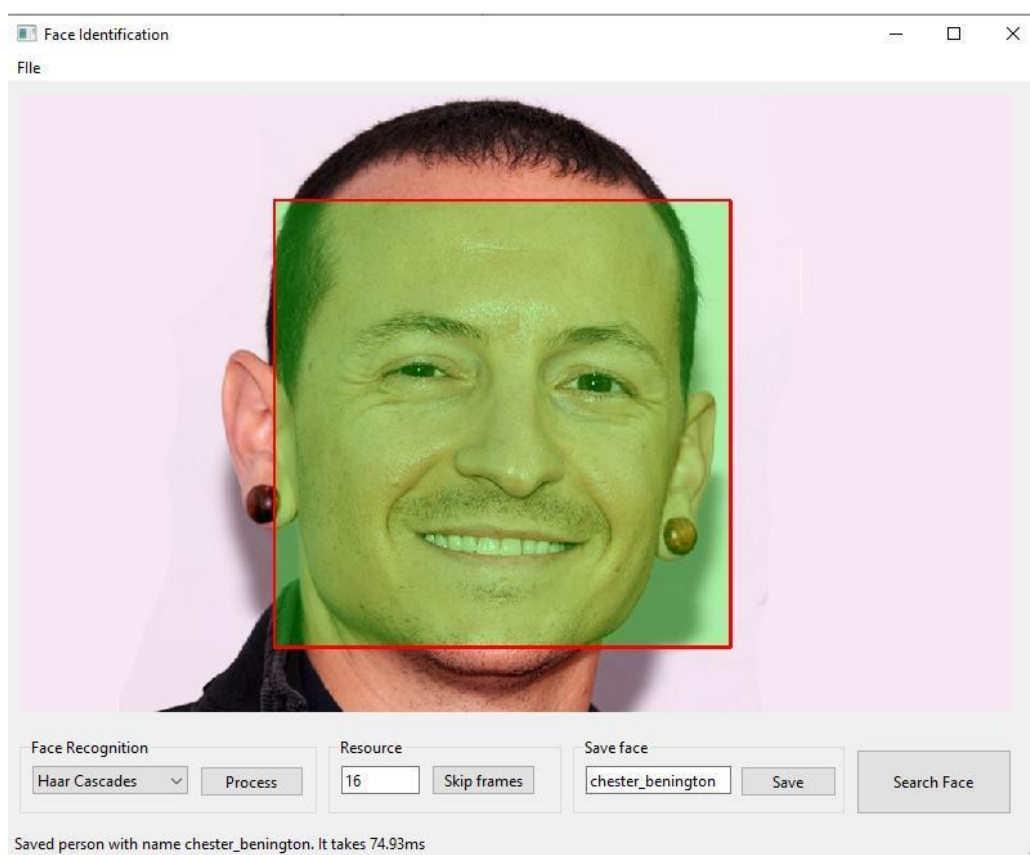


Рисунок 4.3 – Результат зберігання навчальної вибірки

На даному етапі з навчальною вибіркою закінчено. Для наступного етапу роботи потрібна тестова вибірка, а саме завантажити кліп групи Linkin Park - Numb до клієнтського застосунку. Система автоматично розбиває відеофайл на масив зображень, і вже працює окремо з кожним файлом в масиві. Оскільки кліп починається з чорного фрейму, користувач має змогу пропустити деякі фрейми для того що на застосунку відображався бажаний результат. Це можна зробити, кирикиставши кнопку *Skip frames* зі стандартним значенням 16 фреймів.

Вибравши довільний кадр з відео з обличчям Честера, користувач має знову знайти регіон цікавості, натиснувши на *Face Recognition > Process*, вибрати потрібний, та натиснути *Search face* . Клієнтський застосунок відправляє потрібний регіон на сервері отримує результат пошуку. Результат можна поглянути на рисунку 4.4.

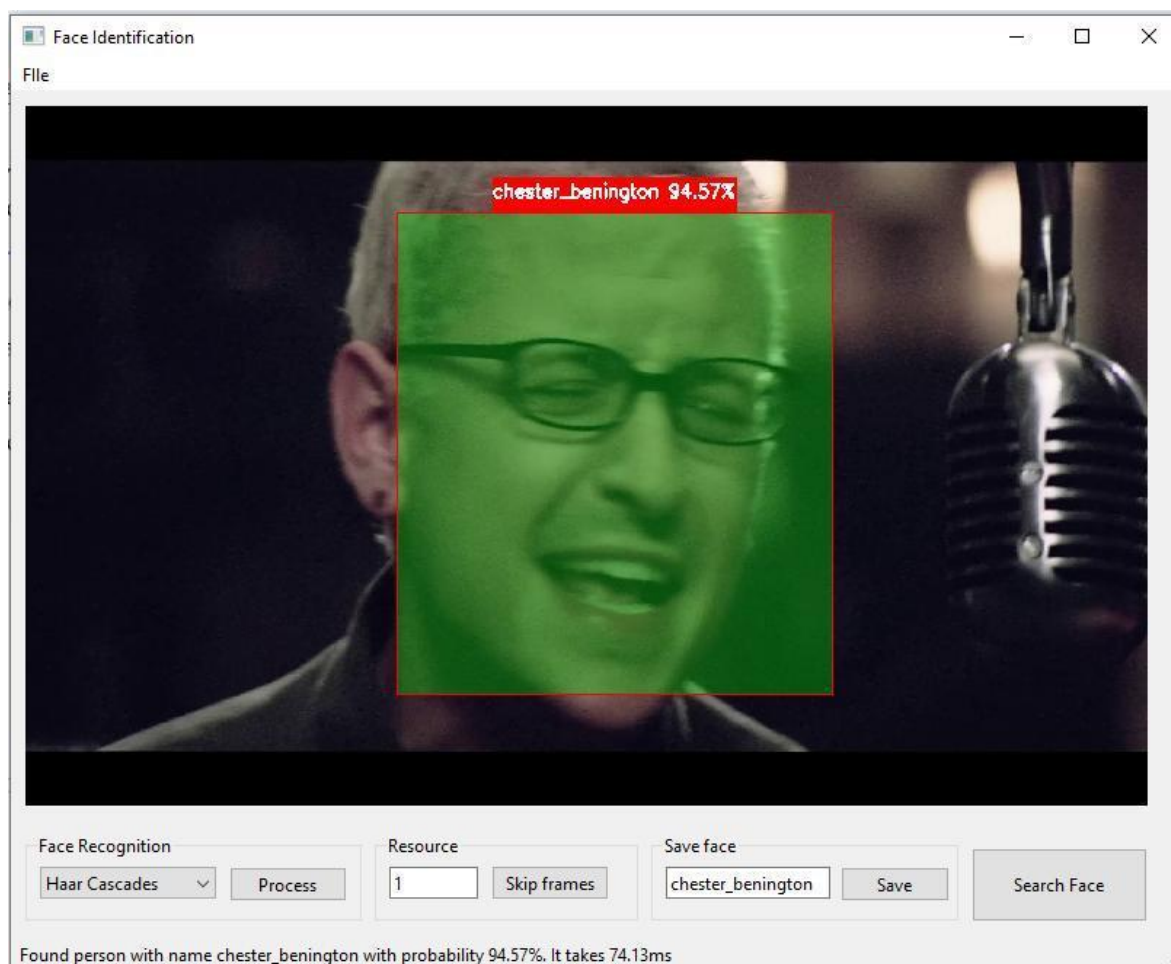


Рисунок 4.4 – Результат пошуку

Як можна побачити на рисунку 4.4, система знайшла співпадіння в базі даних з точністю 94.57%, що є досить непоганим результатом, але оскільки на точність впливають такі фактори як те, що між фото з навчальної вибірки 4 роки різниці (2007 та 2011 роки), також різні вирази обличчя та відсутність окулярів на фото з навчальної вибірки.

Зробивши подібні тести з іншими кадрами в відео, можна спостерігати стабільну роботу системи з різними виразами обличчя та ракурсами (рисунок 4.5) із різними відсотками співпадіння.



Рисунок 4.5 – Результат пошуку

Також система має можливість отримувати навчальні вибірки відразу з тестових відео. Так, головна героїня кліпу, дівчина-художник, теж була занесена до бази, використовуючи кроки, описані вище. Результат пошуку можна побачити на рисунку 4.6, а також аналогічні дії, але вже результати пошуку з солістом групи Imagine Dragons Деном Рейнольдсом на рисунку 4.7.



Рисунок 4.6 – Результат пошуку жінки з тегом *girl_from_numb*



Рисунок 4.7 – Результат пошуку соліста з тегом dan_reynolds

4.2 Аналіз отриманих результатів

Перед тим, щоб почати аналізувати отримані результати розробленої системи, потрібно визначити критерії оцінювання. Раніше в розділах вище зазначалось, що основними показниками роботи системи є швидкість роботи системи та точність розпізнавання, тому розглянемо їх як базові.

Швидкість роботи розробленої в рамках дипломної роботи залежить від декількох основних факторів, які, базуючись на наведений в попередньому розділі діаграмі послідовності, є:

- компресія та пересилання мережею всього фрейму для виділення облич, час в мілісекундах і він може змінюватись в залежності від стану мережі;
- розпізнавання облич - може змінюватись в залежності від кількості облич, вибраного алгоритму та розміру зображення;
- відправка результатів розпізнавання назад клієнту по мережі;
- компресія та пересилання мережею вибраних на клієнтському застосунку регіонів з обличчями на сервер;
- пошук або збереження орієнтирів, обидва механізми включають в себе пошук орієнтирів, у випадку пошуку це ще пошук та порівняння по базі даних, у випадку збереження це вставка в базу даних.

Розглянемо роботу системи на прикладі обробки тестового зображення з п'ятьма персонами різних рас та статей (а саме рисунок 3.5), яке раніше використовувалося для порівняння швидкості алгоритмів, та заміряємо такі величини, як *compression* (швидкість компресії), *network round-trip* (швидкість пересилання мережею туди й назад) та швидкість роботи алгоритму (розпізнавання облич, пошук та збереження). На таблиці 4.1 представлено інформацію про час роботи при використанні функції *Face Recognition*, під терміном *Full frame compression* мається на увазі час компресії всього фрейму (у прикладі – це стискання 1908 кілобайт в 57 кілобайт при якості в 80%)

Таблиця 4.1 – Часові проміжки роботи різних алгоритмів

	Haar Cascades	Dlib HOG	DNN	MediaPipe
Full frame compression	5	5	5	5
Network round-trip	31	30	31	39
Face detection on server	18	14	9	20

На рисунку 4.8 показано порівняльний графік, який базується на даних із таблиці 4.1.

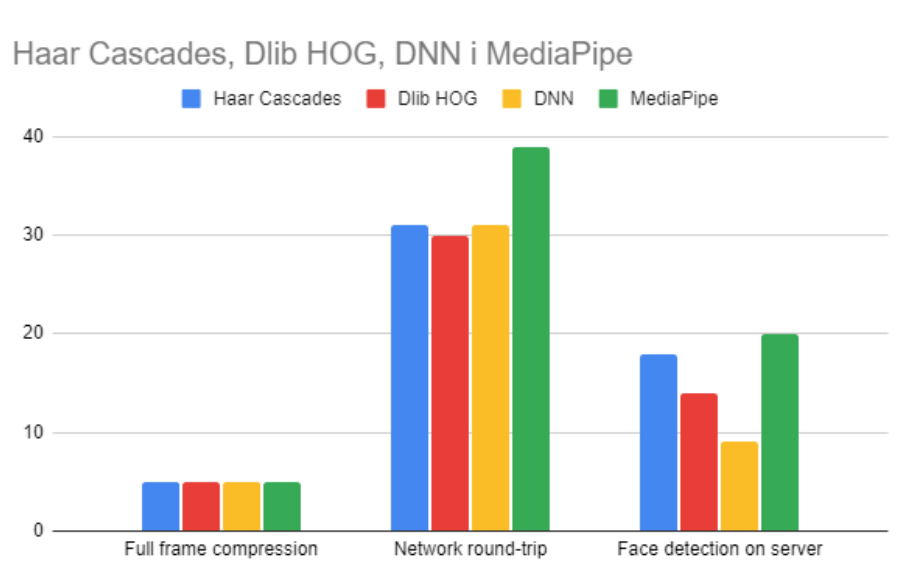


Рисунок 4.8 – Часові проміжки роботи різних алгоритмів

Виходячи з побудованого графіку, вузьким місцем роботи розробленої системи є саме передача фрейму мережею, хоча в цілому показники непогані, бо в цей час входить передача команди протоколу на сервер і відповідь клієнту, передача стиснутого зображення на сервер та відповідь.

Якщо проаналізувати, скільки йде часу на повний цикл розпізнавання (тобто від натискання кнопки на клієнті до відображення клієнтом результатів), то це в середньому 50-60 мілісекунд, що дорівнює 2 кадрам типового відео в 30 frames per second. Це дає змогу плавно програвати цільові відео разом з розпізнаванням облич, при цьому відправляти на сервер кожне друге-третє зображення, досягаючи хороших результатів розпізнавання при плавному відео.

Наступні операції, а саме пошук та збереження не залежать від алгоритмів розпізнавання облич, оскільки робота йде вже з знайденим регіоном, то проаналізуємо середню швидкість повного циклу (клієнт-сервер-клієнт) пошуку та збереження на вже знайденого регіону. На рисунку 4.9 зображений графік ста циклів пошуку орієнтирів, де червоним показано час алгоритму пошуку орієнтирів, а синім - затрати часу на пересилку зображення.

Network round trip i Search

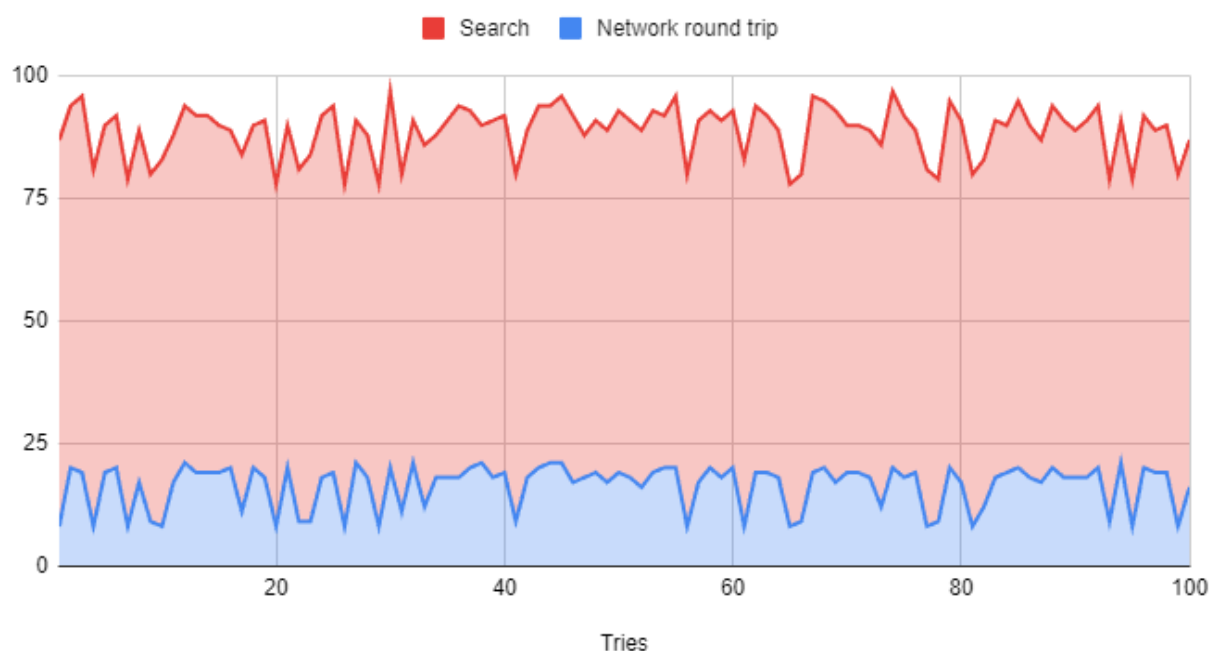


Рисунок 4.9 – Емуляція ста натискань пошуку вибраного обличчя

Середнє значення витрат часу на мережу - 16 мс, та пошук обличчя - 88 мс, що значить середній час роботи дорівнює 104 мс, що доволі таки багато. Проаналізувавши цей результат, можна прийти до висновку, що, хоч мережа і накладає додатковий час до результату, все ж пошук орієнтирів - дуже вартісна операція. Варто додати що це час пошуку лише одного обличчя, тоді як на тестовому зображенні їх п'ять, тобто час пошуку всіх облич збільшується в п'ять раз.

Ще одним вартісним етапом в майбутньому може стати пошук по базі даних, оскільки зі збільшенням числа збережених облич буде збільшуватись і час пошуку, оскільки складність пошуку є лінійною.

Покращити час виконання саме пошуку орієнтирів можна використанням більш потужних графічних відеокарт, наприклад, серію Nvidia RTX 40*, оскільки використані алгоритми орієнтирів добре розпаралелюються.

Оскільки область розпізнавання все ще активно розвивається, і поки що не досягнуто ідеального розпізнавання облич, то іноді можуть виникати так звані false-positive співпадиння – тоді алгоритм спрацьовує на деякі області і помічає їх як обличчя, хоча такими вони не є. На рисунку 4.10 одним з алгоритмів, а саме каскади Хаара, було знайдено false-positive співпадиння на малюнку поруч з героїнею та область на самій героїні, але не її обличчя. В цьому випадку зміна алгоритму наприклад на варіант з глибоким навчанням або на будь-який інший допомогла відсіяти неправильні регіони. В випадку, зображеного на рисунку 4.9, це відкидання всіх знайдених регіонів.



Рисунок 4.10 – Области false-positive регіонів

4.3 Висновки до розділу 4

У цьому розділі було розкрито результати роботи взаємодії серверного та клієнтського застосунків, що було створено за методами та алгоритмами розглянутими в попередніх розділах. Була проаналізована їх робота на прикладі декількох тестових завдань, основною ідеєю яких є ідентифікація людей на фото, а також на відео файлах, конвертованих у масив зображень.

У цілому, всі методи розпізнавання облич показали гарні результати, однак деякі все ж демонстрували некоректні результати, як, наприклад, поява false-positive регіонів у випадку першого тестового відео. Також розглянуто варіанти мінімізації впливу подібних проблем.

Було продемонстровано два варіанти навчання системи, а саме з навчального зображення високої якості при подальшому пошуку людини на цільовому відео, а також коли навчальним зображенням було розпізнане обличчя в тестовому відео. Варто зазначити, що в силу очевидних причин перший варіант є більш точнішим, хоча виходячи з тестів, обидва варіанти навчання були успішними.

Проведений аналіз результатів роботи системи дає досить високу оцінку ефективності роботи в різних умовах, хоча і відчувається вплив поки що невирішених проблем розпізнавання образів, таких як, наприклад, освітленість та наявність false-positive результатів.

ВИСНОВКИ

У дипломному проєкті досліджено методи розпізнавання обличчя за фото та розроблено систему, за допомогою якої можливо ідентифікувати довільне обличчя із вже збережених раніше як на фото, так і на відеокадрах.

У роботі розглянуто основні методи пошуку обличчя на окремо взятому зображенні, виділення ключових орієнтирів на знайденому обличчі, використання орієнтирів як ключових характеристик ідентифікації людини, а також проведено аналіз апаратних рішень для забезпечення ефективної роботи алгоритмів пошуку, таких як GPU та TPU.

Також наведено загальну структуру системи та проведено розділення на ключові підсистеми, в кожену з яких інкапсульовано черговий етап роботи з окремо виділеним зображенням. Ці підсистеми було згруповано окремо в серверний та клієнтський застосунки, і для кожного застосунку було обрано архітектуру та описано етапи розробки. Також було розглянуто важливість використання таких бібліотек, як, наприклад, OpenCV та Dlib, інтерфейс взаємодії з графічною картою CUDA, а також поверхнево наведено вибрані алгоритми для етапів розпізнавання обличчя та виділення орієнтирів. Розглянуто також системи управління базами даних як інструмент зберігання результатів роботи з системою та наведено порівняння тих можливостей PostgreSQL та MySQL, які використовуються в системі.

У дипломному проєкті наводиться практичне застосування використаних в системі бібліотек, таких як DLib, Opencv, Tensorflow та Mediarpipe та стислий огляд моделей глибокого навчання, використаних разом з цими бібліотеками. Для кожного з підходів розпізнавання обличчя проведено порівняння швидкості роботи в різних умовах роботи, а саме коли на зображенні одне, п'ять та десять облич, а також побудовано графік для кожного підходу і кожної умови та на протязі ста ітерацій. Наведено структуру реалізованих серверного та клієнтського застосунків, розроблено та описано структуру класів,

розроблених в рамках дипломного проекту, наведено діаграму послідовностей для серверного застосунку та приведено приклад роботи клієнтського застосунку.

В результаті виконання роботи розроблено систему ідентифікації людини, яка має змогу виділяти обличчя, знаходити унікальні орієнтири, зберігати їх та має змогу шукати по збереженим даним.

Досліджено підходи до розв'язання задачі обличчя на фото та відеокадрах. Визначено доцільність використання класифікаторів та згорткової нейромережі (CNN) у складі бібліотек OpenCV та Dlib. Було висвітлено позитивні та негативні сторони використання саме цього типу нейронних мереж. У роботі розглянуто та застосовано функціонал бібліотеки OpenCV, а саме модулі, які працюють з зображеннями та з моделями глибокого навчання. Розглянуто, як працює вищевказана бібліотека та висвітлені методи, що вона використовує для визначення об'єктів на зображеннях. Також було розглянуто методи, які можна використовувати для покращення якості зображення для отримання кращих результатів її роботи.

У цілому, всі методи розпізнавання облич показали гарні результати, однак деякі все ж демонстрували некорректні результати, як, наприклад, поява false-positive регіонів. Тому було розглянуто варіанти мінімізації впливу подібних проблем.

При тестуванні системи було продемонстровано два варіанти навчання системи, а саме з навчального зображення високої якості при подальшому пошуку людини на цільовому фото/відео, а також коли навчальним зображенням було розпізнане обличчя в тестовому фото/відео. Варто зазначити, що в силу очевидних причин перший варіант є більш точнішим, хоча виходячи з тестів, обидва варіанти навчання були успішними.

Проведений аналіз результатів роботи системи дає досить високу оцінку ефективності роботи в різних умовах, хоча і відчувається вплив поки що невирішених проблем розпізнавання образів, таких як, наприклад, освітленість та наявність false-positive результатів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Forsuh R. Computer Vision: A Modern Approach (2nd Edition) / R. Forsuh, J. Polce - Person, 2011. - 982 p.
2. Applying the Haar-cascade Algorithm for Detecting Safety Equipment in Safety Management Systems for Multiple Working Environments. Le Tran Huu Phuc et al. (2019) URL: <https://www.mdpi.com/2079-9292/8/10/1079> (Last accessed: 05.11.2023).
3. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. Robin M. Schmidt. (2019) URL: <https://arxiv.org/pdf/1912.05911.pdf> (Last accessed: 05.11.2023).
4. Holistic Deep Learning. Dimitris Bertsimas et al. (2023) URL: <https://arxiv.org/pdf/2110.15829.pdf> (Last accessed: 07.11.2023).
5. Comparative Analysis of CPU and GPU Profiling for Deep Learning Models. Dipesh Gyawali. (2023) URL: <https://arxiv.org/pdf/2309.02521.pdf> (Last accessed: 07.11.2023).
6. Training Deep Neural Networks with 8-bit Floating Point Numbers. Naigang Wang. (2018) URL: <https://arxiv.org/pdf/1812.08011.pdf> (Last accessed: 07.11.2023).
7. The Hidden Power of Pure 16-bit Floating-Point Neural Networks. Juyoung Yun, Byungkoo Kang, Zhoulai Fu. (2023) URL: <https://arxiv.org/pdf/2301.12809.pdf> (Last accessed: 07.11.2023).
8. Optimal Architecture of Floating-Point Arithmetic for Neural Network Training Processors. Muhammad Junaid et al. (2022) URL: <https://www.mdpi.com/1424-8220/22/3/1230> (Last accessed: 08.11.2023).
9. RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions. Adeel Ehsan et al. (2022) URL: <https://www.mdpi.com/2076-3417/12/9/4369> (Last accessed: 11.11.2023).
10. OpenCV. URL: <https://opencv.org/> (Last accessed: 07.12.2023).

11. DEEP NEURAL NETWORKS FOR PATTERN RECOGNITION. Kyongsik Yun, PhD, Alexander Huyen, Thomas Lu. (2018) URL: <https://arxiv.org/ftp/arxiv/papers/1809/1809.09645.pdf> (Last accessed: 12.11.2023).
12. TensorFlow. URL: <https://www.tensorflow.org/guide> (Last accessed: 15.11.2023).
13. Histogram of Oriented Gradient-Based Fusion of Features for Human Action Recognition in Action Video Sequences. Chirag I. Patel. (2020) URL: <https://www.mdpi.com/1424-8220/20/24/7299> (Last accessed: 15.11.2023).
14. MediaPipe: A Framework for Building Perception Pipelines. Camillo Lugaresi. (2019) URL: <https://arxiv.org/pdf/1906.08172.pdf> (Last accessed: 15.11.2023).
15. MobileNetV2: Inverted Residuals and Linear Bottlenecks. Mark Sandler et al. (2019) URL: <https://arxiv.org/pdf/1801.04381.pdf> (Last accessed: 15.11.2023).
16. Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Network. Kaipeng Zhang et al. (2016) URL: <https://arxiv.org/ftp/arxiv/papers/1604/1604.02878.pdf> (Last accessed: 17.11.2023).
17. Mariia Tiahunova, Olesia Tronkina, Galina Kirichek, Stepan Skrupsky. The Neural Network for Emotions Recognition under Special Conditions. In CMIS-2021: The Third International Workshop on Computer Modeling and Intelligent Systems, Zaporizhzhia, Ukraine, April 27, 2021. p. 121-134, ISSN 1616-0073
18. Б.М. Бобирь, М.Ю. Тягунова. Ідентифікація людини на відео в реальному часі. Тиждень науки-2023. Факультет комп'ютерних наук і технологій : наук.-техн. конф., 24-28 квітня 2023 р. : тези доп.. – Запоріжжя: НУ «Запорізька політехніка», 2023. С. 61-62.