Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента *Корнієнка Дмитра Ігоровича*
(ПІБ)

академічної групи *122М-19-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

на тему *Методи, алгоритми та інформаційна технологія розпізнавання обличчя на основі згорткової нейронної мережі*

Д.І.Корнієнко

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
розділів кваліфікаційної роботи				
спеціальний	Проф. Алексєєв М.О.			
економічний	Доц. Касьяненко Л.В.			

Рецензент				

Нормоконтролер	Доц. Сироткіна О.І.			

Дніпро
2020

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)
І.М. Удовик
(підпис) (прізвище, ініціали)
« » 20 20 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)
студенту *122м-19-1* (група) *Корнієнку Дмитру Ігоровичу* (прізвище та ініціали)
Тема кваліфікаційної роботи *Методи, алгоритми та інформаційна технологія розпізнавання обличчя на основі згорткової нейронної мережі*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 22.10.2020 р. № 888-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процес розпізнавання обличчя в відеопотоках у режимі реального часу.

Предмет досліджень – методи розпізнавання обличчя за допомогою нейронних мереж.

Мета роботи – вдосконалення та пришвидшення процесу розпізнавання обличчя в відеопотоках в режимі реального часу з використанням нейронних мереж.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна полягає в удосконаленні та оптимізації методу розпізнавання обличчя за допомогою нейронних мереж та засобів програмування JavaScript.

Практична цінність полягає в тому, що розроблена програма буде корисна в області систем безпеки для контролю пропуску осіб на певну територію а також в області систем відеоспостереження.

4 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	12.09.2020-20.10.2020
Розробка комп'ютерної системи з розпізнавання обличчя на основі нейронних мереж, з метою вдосконалення процесів розпізнавання обличчя.	15.10.2020-10.11.2020
Оптимізація системи та її влаштування у веб-інтерфейс.	11.11.2020-10.12.2020

Завдання видав _____ *Алексєєв М.О.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Корнієнко Д.І.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *20.10.2020 р.*

Термін подання кваліфікаційної роботи до ЕК *14.12.2020*

РЕФЕРАТ

Пояснювальна записка: 68 с., 27 рис., 4 дод., 28 джерел.

Об'єкт дослідження: процес розпізнавання обличчя в відеопотоках у режимі реального часу.

Предмет дослідження: методи розпізнавання обличчя за допомогою нейронних мереж.

Мета кваліфікаційної роботи: вдосконалення та пришвидшення процесу розпізнавання обличчя в відеопотоках в режимі реального часу з використанням нейронних мереж.

Наукова новизна полягає в удосконаленні та оптимізації методу розпізнавання обличчя за допомогою нейронних мереж та засобів програмування JavaScript.

Практичне значення роботи. Розроблена програма буде корисна в області систем безпеки для контролю пропуску осіб на певну територію а також в області систем відеоспостереження.

У розділі «Економіка» проведено розрахунки трудомісткості розробки програмного забезпечення, витрат на створення ПО і тривалості його розробки.

Список ключових слів: згорткова нейронна мережа, розпізнавання обличчя, режим реального часу, розпізнавання образів, javascript.

ABSTRACT

Explanatory note: 68 p., 27 fig., 4 applications, 28 sources. .

Object of research: the process of face recognition in video streams in real time.

Subject of research: methods of facial recognition using neural networks.

The purpose of the qualification work: to improve and accelerate the process of face recognition in video streams in real time using neural networks**.**

The scientific novelty is to improve and optimize the method of facial recognition using neural networks and JavaScript programming tools.

The practical significance of the work. The developed program will be useful in the field of security systems to control the passage of persons to a certain area, as well as in the field of video surveillance systems.

In the section "Economics" calculations of the complexity of software development, the cost of creating software and the duration of its development.

Keyword list: rolled neural network, face recognition, real time mode, image recognition, javascript.

ЗМІСТ

ВСТУП

Актуальність дослідження. В даний час для ідентифікації людини використовуються біометричні методи. Біометрією називається сукупність способів і пристроїв для ідентифікації людини, які засновані на його унікальних фізіологічних або поведінкових характеристиках. У зв'язку з простотою розпізнавання особи і великою кількістю камер у всіх аспектах життя людини, все більш актуальним стають розробки в розпізнаванні осіб. Завдання розпізнавання осіб актуальна як в області інтелектуальних середовищ, так і в системах безпеки.

Розпізнавання обличчя можна розбити на три пункти:

1) знайти особу в режимі реального часу;

2) порівняти знайдену особу з особами, що зберігаються в базі даних;

3) порівняти знайдену особу з еталонним в базі даних.

Але якщо людина може визначити і порівняти особу з особами, що зберігаються в пам'яті, за частки секунди, то без належної технології розпізнавання машина не зможе відрізнити людину від стовпа. Людина впізнає знайоме обличчя, орієнтуючись на індивідуальні риси, а саме відстань між очей, їх колір, висота губ і їх ширина. Для початку комп'ютер повинен не просто розпізнати людину, але і зрозуміти, що знаходиться перед ним, особа чи це або ваза. При тому ракурс, з якого камера приймає обличчя людини, гра світла або ж зайві предмети на обличчі людини відіграють величезну роль.

В даний час спостерігається неминущий інтерес до проблеми розпізнавання обличчя. Розпізнавання обличчя можна умовно поділити на 2 завдання: ідентифікація і верифікація. Завдання ідентифікації полягає в тому, щоб порівняти захоплене обличчя з усіма зображеннями осіб, що зберігаються в базі даних. Мета верифікації полягає в тому, щоб порівняти обличчя людини з фотографії кандидата, з особою на еталонної фотографії, що зберігається в базі даних.

Завдання розпізнавання осіб актуальна як в області інтелектуальних

середовищ, так і в системах безпеки. Наприклад, в цьому році був запущений онлайн-сервіс розпізнавання осіб з фотографій в інтернеті на основі російської технології, розробленої N-Tech.Lab. Google також опублікував наукову роботу про нову систему штучного інтелекту FaceNet, яка розпізнає обличчя людей з точністю 99,63% на стандартному наборі даних LFW. Система дослідників з Facebook показала результат близько 97,5%.

Мета дослідження: вдосконалення та пришвидшення процесу розпізнавання обличчя в відеопотоках в режимі реального часу з використанням нейронних мереж.

Завдання дослідження: для досягнення цілі потрібно виконати наступні завдання:

1) Проаналізувати існуючі підходи для розпізнавання осіб;

2) Виявити класифікацію алгоритмів розпізнавання;

3) Провести аналіз існуючих на ринку систем розпізнавання,
виявити їхні переваги і недоліки;

4) Проаналізувати основні інструментальні засоби для
розробки і вибрати оптимальні з них;

5) Спроектувати і програмно реалізувати систему розпізнавання.

Об’єкт дослідження: процес розпізнавання обличчя в відеопотоках у режимі реального часу.

Предмет дослідження: методи розпізнавання обличчя за допомогою нейронних мереж.

Методи дослідження: Для створення комп’ютерної системи були використані засоби та методи розпізнавання обличчя, бібліотека FaceAPI і деякі функції загорткової нейронної мережі Mobilnetv1.

Особистий внесок автора:

1. Описані принципи роботи нейронних мереж у задачах з розпізнавання обличчя.

2. Було проаналізовано та порівняно різні методи розпізнавання.

3. Було створено програму, що демонструє успішне та швидке розпізнавання обличчя і потребує при цьому мінімальні ресурси системи.

Наукова новизна: вдосконалення та пришвидшення процесу розпізнавання обличчя в відеопотоках в режимі реального часу з використанням нейронних мереж.

Практичне значення. Розроблена програма буде корисна в області систем безпеки для контролю пропуску осіб на певну територію а також в області систем відеоспостереження.

Структура і обсяг роботи. Робота складається з вступу, чотирьох розділів і висновків. Містить 68 сторінок, в тому числі 59 сторінок тексту основної частини з 27 рисунками, списку використаних джерел з 28 найменуваннями на 4 сторінках, 3 додатка на 9 сторінках.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ВІДОМОСТІ ТА АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕНЬ

1.1. Існуючі засоби розпізнавання обличчя

Незважаючи на велику різноманітність представлених алгоритмів, можна виділити загальну структуру процесу розпізнавання обличчя:

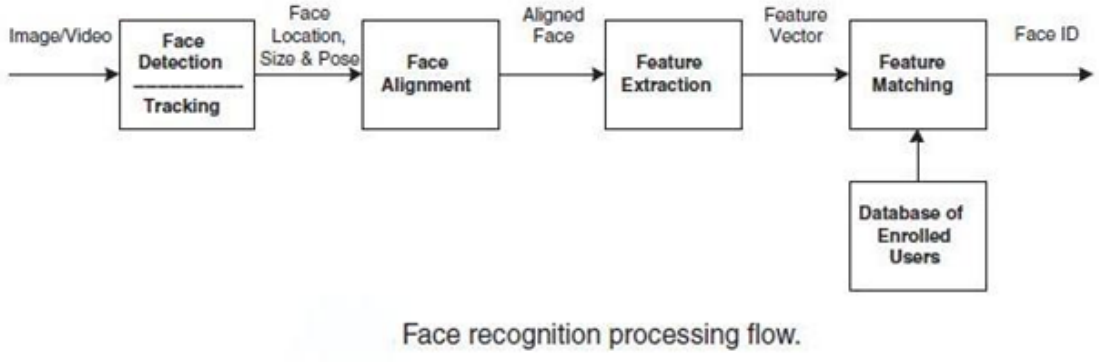


Рис 1.1. Загальний процес обробки зображення обличчя при розпізнаванні

На першому етапі проводиться детектування і локалізація обличчя на зображенні. На етапі розпізнавання проводиться вирівнювання зображення обличчя (геометричне і яскравістне), обчислення ознак і безпосередньо розпізнавання - порівняння обчислених ознак з закладеними в базу даних еталонами. Основною відмінністю всіх представлених алгоритмів буде обчислення ознак і порівняння їх сукупностей між собою.

1.1.1. Метод гнучкого порівняння на графах

Суть методу зводиться до еластичного порівняння графів, що описують зображення обличчя. Особи представлені у вигляді графів зі зваженими вершинами і ребрами. На етапі розпізнавання один з графів - еталонний – залишається незмінним, в той час як інший деформується з метою найкращої підгонки до першого. У подібних системах розпізнавання графи можуть являти

собою як прямокутну решітку, так і структуру, утворену характерними (антропометричними) точками особи.

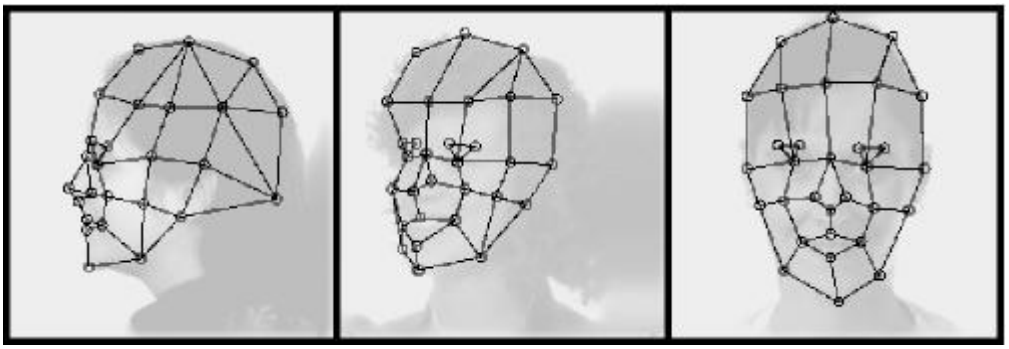


Рис 1.2. Граф на основі антропометричних точок обличчя

У вершинах графа обчислюються значення ознак, найчастіше використовують комплексні значення фільтрів Габора або їх упорядкованих наборів – Габорівських вейвлет (строї Габора), які обчислюються в деякій локальній області вершини графа локально шляхом згортки значень яскравості пікселів з фільтрами Габора.

Ребра графа зважуються відстанями між суміжними вершинами. Різниця (відстань, дискримінаційна характеристика) між двома графами обчислюється за допомогою деякої цінової функції деформації, що враховує як відмінність між значеннями ознак, обчисленими в вершинах, так і ступінь деформації ребер графа.

Деформація графа відбувається шляхом зсуву кожної з його вершин на деяку відстань в певних напрямках щодо її вихідного розташування і вибору такої її позиції, при якій різниця між значеннями ознак (відгуків фільтрів Габора) в вершині деформованого графа і відповідної їй вершині еталонного графа буде мінімальною. Дана операція виконується по черзі для всіх вершин графа до тих пір, поки не буде досягнуто найменша сумарна відмінність між ознаками деформованого і еталонного графів. Значення цінової функції деформації при такому положенні деформуючого графа і буде мірою відмінності між вхідним зображенням обличчя і еталонним графом. Дана

«релаксаційна» процедура деформації повинна виконуватися для всіх еталонних осіб, закладених в базу даних системи. Результат розпізнавання системи - еталон з найкращим значенням цінової функції деформації.

1.1.2. Нейронні мережі

В даний час існує близько десятка різновиди нейронних мереж (НМ). Один з найбільш широко використовуваних варіантів це мережа, побудована на багатошаровому перцептроні, яка дозволяє класифікувати подане на вхід зображення / сигнал відповідно до попередніх налаштувань / навчань мережі.

Навчаються нейронні мережі на наборі навчальних прикладів. Суть навчання зводиться до налаштування ваг міжнейронних зв'язків в процесі рішення оптимізаційної задачі методом градієнтного спуску. В процесі навчання НМ відбувається автоматичне вилучення ключових ознак, визначення їх важливості та побудова взаємозв'язків між ними. Передбачається, що навчена НМ зможе застосувати досвід, отриманий в процесі навчання, на невідомі образи за рахунок узагальнюючих здібностей.

Найкращі результати в області розпізнавання осіб (за результатами аналізу публікацій) показала Convolutional Neural Network або згорткова нейронна мережа (далі - ЗНМ), яка є логічним розвитком ідей таких архітектур НМ як когнітрону і неокогнітрона. Успіх обумовлений можливістю обліку двовимірної топології зображення, на відміну від багатошарового перцептрона.

Відмінними рисами ЗНМ є локальні рецепторні поля (забезпечують локальну двовимірну зв'язність нейронів), загальні ваги (забезпечують детектування деяких рис в будь-якому місці зображення) і ієрархічна організація з просторовим семплінгом (spatial subsampling). Завдяки цим нововведенням ЗНМ забезпечує часткову стійкість до змін масштабу, зсувів, поворотів, зміні ракурсу і інших спотворень.

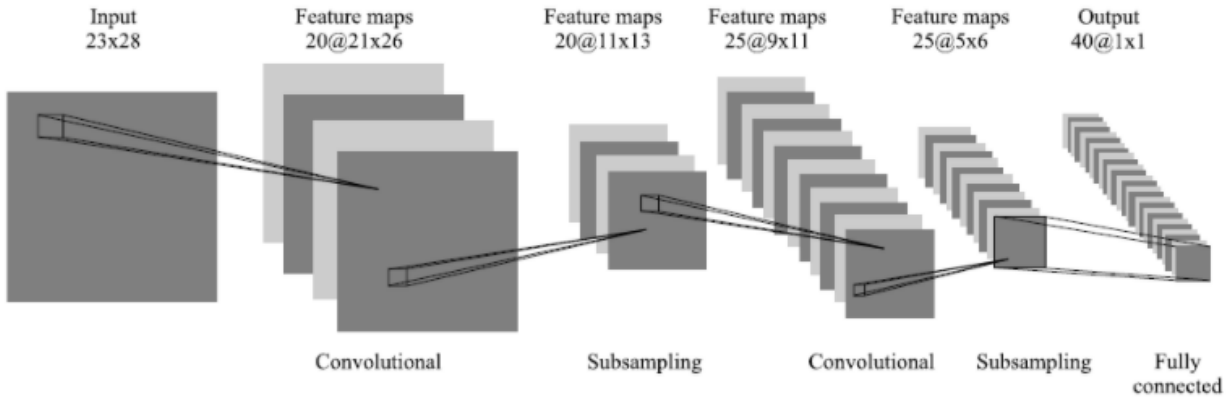


Рис 1.3. Схематичне зображення архітектури згорткової нейронної мережі

Тестування ЗНМ на базі даних ORL, що містить зображення осіб з невеликими змінами освітлення, масштабу, просторових поворотів, положення і різними емоціями, показало 96% точність розпізнавання.

Свій розвиток ЗНМ отримали в розробці DeepFace , яку придбав Facebook для розпізнавання осіб користувачів своєї соцмережі. Всі особливості архітектури носять закритий характер.

Недоліки нейронних мереж: додавання нової еталонної особи в базу даних вимагає повного перенавчання мережі на всьому наявному наборі (досить тривала процедура, в залежності від розміру вибірки від 1 години до декількох днів). Проблеми математичного характеру, пов'язані з навчанням: потрапляння в локальний оптимум, вибір оптимального кроку оптимізації, перенавчання і т.д. Важко формалізується етап вибору архітектури мережі (кількість нейронів, шарів, характер зв'язків). Узагальнюючи все вищесказане, можна зробити висновок, що НМ - «чорний ящик» з важко інтерпретуючими результатами роботи.

1.1.3. Приховані Марківські моделі (ПММ)

Одним з статистичних методів розпізнавання обличчя є приховані Марківські моделі (ПММ) з дискретним часом. ПММ використовують статистичні властивості сигналів і враховують безпосередньо їх просторові

характеристики. Елементами моделі є: безліч прихованих станів, безліч спостережуваних станів, матриця перехідних ймовірностей, початкова ймовірність станів. Кожному відповідає своя Маркіська модель. При розпізнаванні об'єкта перевіряються згенеровані для заданої бази об'єктів Марківські моделі і шукається максимальна із спостережуваних ймовірність того, що послідовність спостережень для даного об'єкта згенерована відповідною моделлю.

На сьогоднішній день не вдалося знайти приклад комерційного застосування ПММ для розпізнавання осіб.

Недоліки:

- необхідно підбирати параметри моделі для кожної бази даних;
- ПММ не володіє розрізняючою здатністю, тобто алгоритм навчання тільки максимізує відгук кожного зображення на свою модель, але не мінімізує відгук на інші моделі.

1.1.4. Метод головних компонент або principal component analysis (PCA)

Одним з найбільш відомих і опрацьованих є метод головних компонент (principal component analysis, PCA), заснований на перетворенні Карунена-Лоєва.

Спочатку метод головних компонент почав застосовуватися в статистиці для зниження простору ознак без істотної втрати інформації. У задачі розпізнавання осіб його застосовують головним чином для представлення зображення особи вектором малої розмірності (головних компонент), який порівнюється потім з еталонними векторами, закладеними в базу даних.

Головною метою методу головних компонент є значне зменшення розмірності простору ознак таким чином, щоб воно якомога краще описувало «типові» образи, що належать безлічі осіб. Використовуючи цей метод можна виявити різні мінливості в навчальній вибірці зображень облич і описати цю мінливість в базисі декількох ортогональних векторів, які називаються

власними (eigenface).

Отриманий один раз на навчальній вибірці зображень облич набір власних векторів використовується для кодування всіх інших зображень осіб, які представляються зваженої комбінацією цих власних векторів. Використовуючи обмежену кількість власних векторів можна отримати стислу апроксимацію вхідному зображенню особи, яку потім можна зберігати в базі даних у вигляді вектора коефіцієнтів, службовця одночасно ключем пошуку в базі даних осіб.

Суть методу головних компонент зводиться до наступного. Спочатку весь навчальний набір осіб перетвориться в одну загальну матрицю даних, де кожен рядок являє собою один екземпляр зображення особи, розкладеного в рядок. Всі особи навчального набору повинні бути приведені до одного розміру і з нормованими гістограмами.

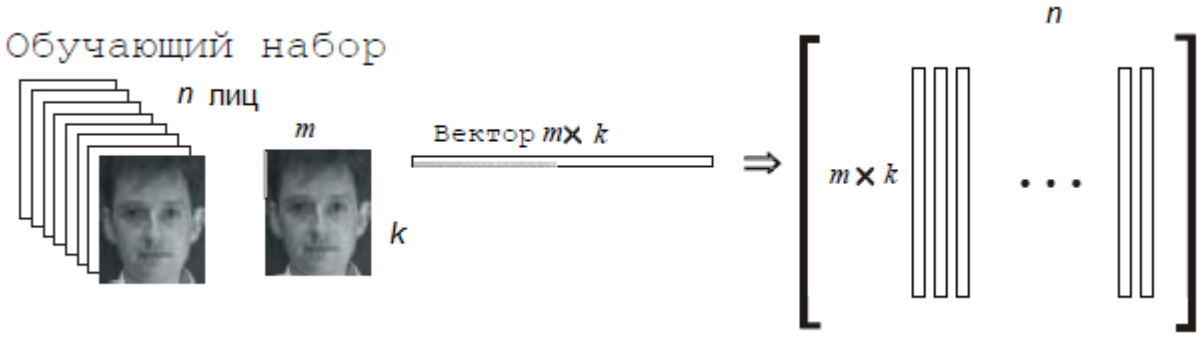


Рис 1.4. Перетворення навчального набору осіб в одну загальну матрицю X

Метод головних компонент добре зарекомендував себе в практичних додатках. Однак, в тих випадках, коли на зображенні особи присутні значні зміни в освітленості або виразі обличчя, ефективність методу значно падає. Вся справа в тому, що PCA вибирає підпростір з такою метою, щоб максимально апроксимувати вхідний набір даних, а не виконати дискримінацію між класами осіб.

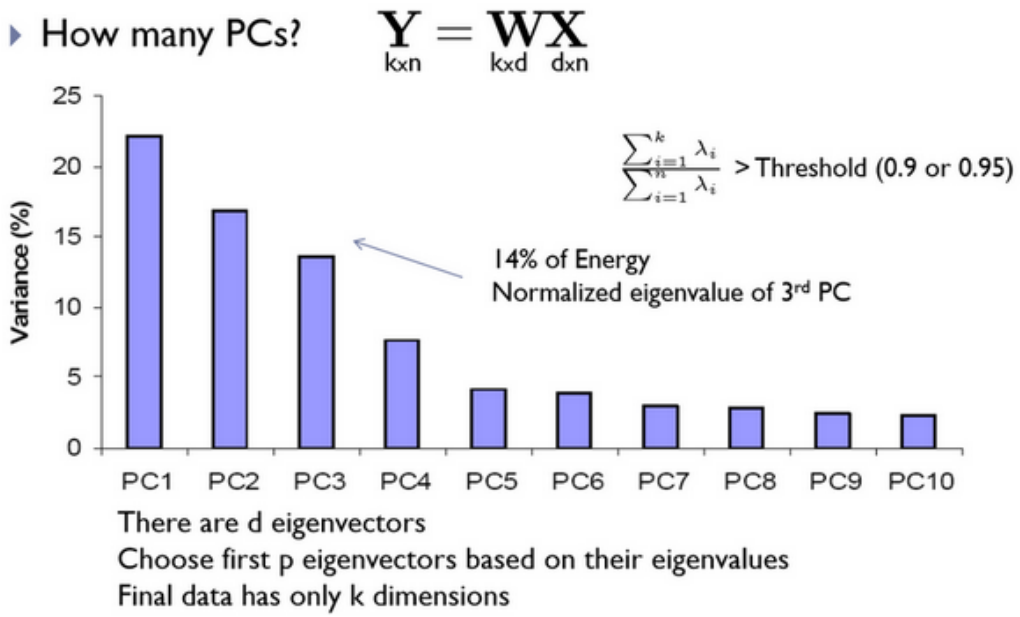


Рис 1.5. Принцип вибору базису з перших кращих власних векторів

1.1.5. Active Appearance Models (AAM) і Active Shape Models (ASM)

Active Appearance Models (AAM)

Активні моделі зовнішнього вигляду (Active Appearance Models, AAM) - це статистичні моделі зображень, які шляхом різного роду деформацій можуть бути підігнані під реальне зображення. Даний тип моделей в двовимірному варіанті було запропоновано Тімом Кутса і Крісом Тейлором в 1998 році. Спочатку активні моделі зовнішнього вигляду застосовувалися для оцінки параметрів зображень облич.

Активна модель зовнішнього вигляду містить два типи параметрів: параметри, пов'язані з формою (параметри форми), і параметри, пов'язані зі статистичною моделлю пікселів зображення або текстурою (параметри зовнішнього вигляду). Перед використанням модель повинна бути навчена на безлічі заздалегідь розмічених зображень. Розмітка зображень виробляється вручну. Кожна мітка має свій номер і визначає характерну точку, яку повинна буде знаходити модель під час адаптації до нового зображення.

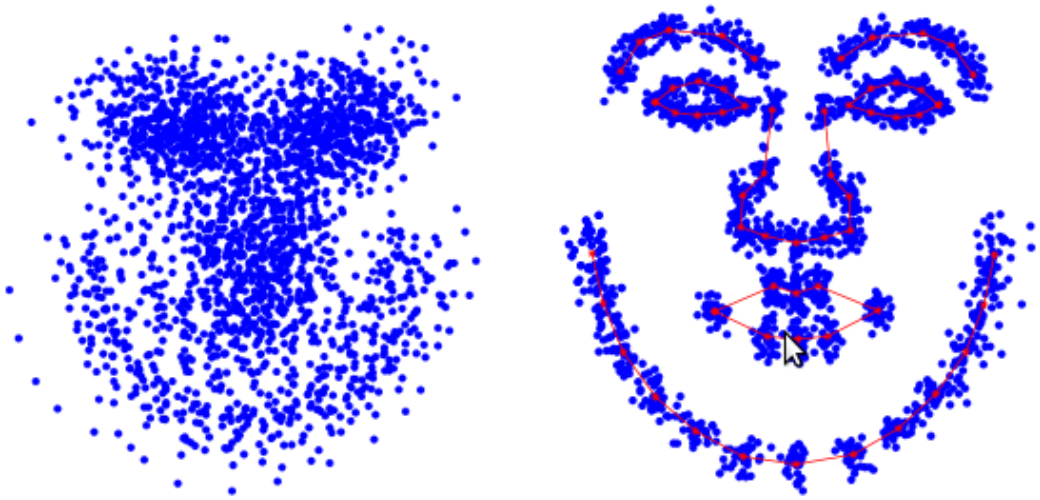


Рис 1.6. Координати точок форми обличчя до і після нормалізації

З усієї безлічі нормованих точок потім виділяються головні компоненти з використанням методу PCA.

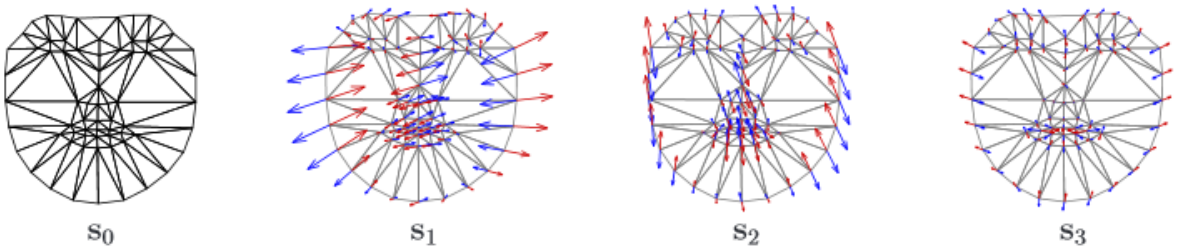


Рис 1.7. Модель форми AAM складається з триангуляційної решітки s0 і лінійної комбінації зсувів si щодо s0

Далі з пікселів всередині трикутників, утворених точками форми, формується матриця, така що, кожен її стовпець містить значення пікселів відповідної текстури. Варто відзначити, що використовувані для навчання текстури можуть бути як одноканальними (градації сірого), так і багатоканальними (наприклад, простір кольорів RGB або інше). У разі багатоканальних текстур вектори пікселів формуються окремо по кожному зканалів, а потім виконується їх конкатенація. Після знаходження головних компонент матриці текстур модель AAM вважається навченої.

Модель зовнішнього вигляду AAM складається з базового виду A0, визначеного пікселями всередині базової решітки s0 і лінійної комбінації зсувів

Ai щодо A0. Приклад конкретизації AAM. Вектор параметрів форми:
p = (p_1, p_2, ⟦..., p⟧ _m) ^ T = (- 54,10, -9.1, ...) ^ T використовується для синтезу моделі форми s, а вектор параметрів λ = (λ_1, λ_2, ⟦..., λ⟧ _m) ^ T = (3559,351, -256, ...) ^ Тдля синтезу зовнішнього вигляду моделі. Підсумкова модель особи ⟦M (W (x; p))⟧ ^ виходить як комбінація двох моделей - форми і зовнішнього вигляду.

Підгонка моделі під конкретне зображення обличчя виконується в процесі рішення оптимізаційної задачі, суть якої зводиться до мінімізації функціоналу. $p = (p_1, p_2, \dots, p_m)^T = (-54{,}10, -9.1, \dots)^T$
методом градієнтного спуску. Знайдені при цьому параметри моделі і будуть відображати положення моделі на конкретному зображенні.

За допомогою AAM можна моделювати зображення об'єктів, схильних до як жорсткої, так і нежорсткій деформації. AAM складається з набору параметрів, частина яких представляють форму особи, інші задають його текстуру. Під деформації зазвичай розуміють геометричне перетворення у вигляді композиції перенесення, повороту і масштабування. При вирішенні задачі локалізації особи на зображенні виконується пошук параметрів (розташування, форма, текстура) AAM, які представляють синтезуюче зображення, найбільш близьке до спостережуваного. За ступенем близькості AAM підганяти зображення приймається рішення - є обличчя чи ні.

Active Shape Models (ASM)

Суть методу ASM полягає в обліку статистичних зв'язків між розташуванням антропометричних точок. На наявної вибірці зображень осіб, знятих в анфас. На зображенні експерт розмічає розташування антропометричних точок. На кожному зображенні точки пронумеровані в однаковому порядку.

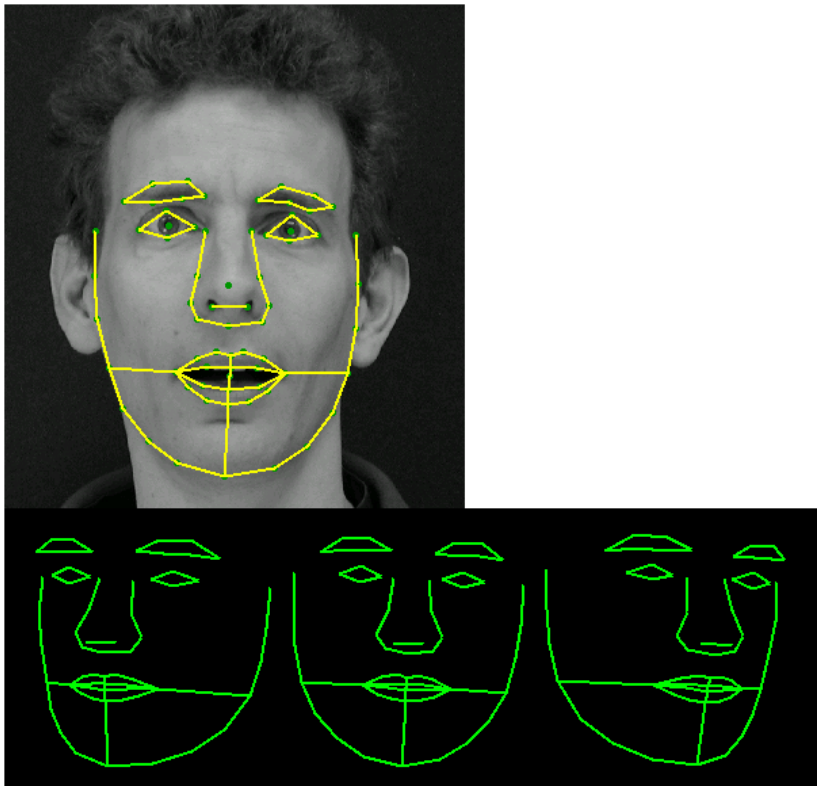


Рис 1.8. Приклад подання форми обличчя з використанням 68 точок

Для того щоб привести координати на всіх зображеннях до єдиної системи зазвичай виконується т.зв. узагальнений Прокрустом аналіз, в результаті якого всі точки приводяться до одного масштабу і центруються. Далі для всього набору образів обчислюється середня форма і матриця коваріації. На основі матриці коваріації обчислюються власні вектора, які потім сортуються в порядку убування відповідних їм власних значень. Модель ASM визначається матрицею Ф і вектором середньої форми $\bar{s}$.

Тоді будь-яка форма може бути описана за допомогою моделі і параметрів:

$$b_i = \Phi^T \bar{s}_i = \Phi^T (s_i - \bar{s})$$

Локалізації ASM моделі на новому, що не входить в навчальну вибірку зображенні здійснюється в процесі рішення оптимізаційної задачі.

Однак все ж головною метою AAM і ASM є не розпізнавання осіб, а точна локалізація особи і антропометричних точок на зображенні для подальшої обробки.

Практично у всіх алгоритмах обов'язковим етапом, що передує класифікацію, є вирівнювання, під яким розуміється вирівнювання зображення

особи у фронтальне положення щодо камери або приведення сукупності осіб (наприклад, в навчальній вибірці для навчання класифікатора) до єдиної системи координат. Для реалізації цього етапу необхідна локалізація на зображенні характерних для всіх осіб антропометричних точок - найчастіше це центри зіниць або куточки очей. Різні дослідники виділяють різні групи таких точок. З метою скорочення обчислювальних витрат для систем реального часу розробники виділяють не більше 10 таких точок .

Моделі AAM і ASM якраз і призначені для того щоб точно локалізувати ці антропометричні точки на зображенні особи.

1.1.6. Алгоритм Віоли-Джонса

В основу методу Віоли-Джонса покладені: інтегральне представлення зображення за ознаками Хаара, побудова класифікатора на основі алгоритму адаптивного бустингу і спосіб комбінування класифікаторів в каскадну структуру. Даний метод демонструє високу ефективність при вирішенні завдання пошуку об'єктів на зображеннях і відеопослідовність в режимі реального часу. Алгоритм Віоли-Джонса має низьку ймовірністю помилкового виявлення обличчя. Метод дозволяє виявляти обличчя при спостереженні його під кутом до 30 °. Точність ідентифікації може досягати значень понад 90%. Метод був розроблений в 2001 році, має велику кількість реалізацій і широко застосовується на практиці, як простий і ефективний. Алгоритм Віоли-Джонса має реалізацію в вільно розповсюджуваної бібліотеці OpenCV, що дозволяє використовувати даний алгоритм в розроблюваної системі відеоспостереження [13].

Метод гнучкого порівняння на графах має великий час розпізнавання і обчислювальну складність. Метод головних компонент має більший, в порівнянні з іншими варіантами, вплив змін міміки на точність розпізнавання. За результатами порівняльного аналізу найбільш відповідними алгоритмами для розпізнавання осіб в програмній системі ситуаційного відеоспостереження

є нейронні мережі і метод Віоли-Джонса. Обидва алгоритми задовольняють умовам технічного завдання і мають високі показники ефективності. Однак, метод нейронних мереж, на відміну від алгоритму Віоли-Джонса, не має реалізацій на мовах Delphi, C, C ++, C #, які вказані в якості бажаної для реалізації проекту. Вибір між використанням нейромереж і методу Віоли-Джонса був зроблений на користь останнього, так як він має реалізацію засобами вільної бібліотеки комп'ютерного зору OpenCV. Робота з нею можлива на мові C ++, а завдяки оболонці EmguCV - також на мові C #.

1.2. Основні проблеми, пов'язані з розробкою систем розпізнавання осіб

Проблема освітленості

Одна з основних проблем в задачі розпізнавання осіб пов'язана з впливом освітленості на якість розпізнавання. Більшість алгоритмів розпізнавання осіб успішно справляються зі своїм завданням, коли все зображення осіб отримані при однакових умовах освітлення. Якщо колекція містить зображення, отримані при різних умовах зйомки, наприклад, в приміщенні і на вулиці, це являє істотну складність, і алгоритми розпізнавання осіб вважають фотографії, отримані при подібному освітленні, схожими, а що не фотографії одного і того ж людини, отримані при різних умовах.

Проблема положення голови

Вибір вихідного опису осіб, що піддаються подальшому аналізу і розпізнавання, є однією з центральних завдань проблеми розпізнавання та ідентифікації. При вдалому виборі вихідного опису (простору ознак) завдання розпізнавання може виявитися тривіальної і, в той же час, невдало вибране вихідне опис може привести або до дуже складної подальшій обробці даних, або взагалі до відсутності рішення[11] .

Так, наприклад, при скануванні особи анфас в приміщенні з відмінною освітленістю хороші результати демонструють алгоритми, що працюють з двомірними зображеннями. Аналізуючи унікальні точки і відстані між ними,

система розпізнавання облич визначає факт ідентифікації за коефіцієнтами відмінності між «живим» знімком і зареєстрованим шаблоном.

Тривимірні технології стійкі до зміни світлового потоку, допустиме відхилення від фронтального ракурсу - до 45 градусів. Тут аналізу піддаються не тільки точки і лінії, а й властивості поверхонь (кривизна, профіль), метрика відстаней між ними.

Однак, для роботи таких алгоритмів, потрібна обробка відеозапису високої якості з частотою не менш 200 кадрів в секунду, обумовлена не тільки високою оптичною роздільною здатністю відеокамер і зведеної до мінімуму похибкою синхронізації, але і високою продуктивністю систем розпізнавання.

Таким чином, основною проблемою, пов'язаною з розпізнаванням і ідентифікацією осіб по відеофіксації в режимі реального часу, є необхідність зниження рівня незалежності роботи системи від факторів освітленості, ракурсу і вікових змін осіб поряд з обчислювальними і продуктивними вимогами до використовуваних в системі пристроїв [12].

З метою оцінки ефективності запропонованих алгоритмів розпізнавання осіб агентство DARPA і дослідницька лабораторія армії США розробили програму FERET (face recognition technology).

У масштабних тестах програми FERET брали участь алгоритми, засновані на гнучкому порівнянні на графах і всілякі модифікації методу головних компонент (PCA). Ефективність всіх алгоритмів була приблизно однаковою. У зв'язку з цим важко або навіть неможливо провести чіткі відмінності між ними (особливо якщо узгодити дати тестування). Для фронтальних зображень, зроблених в один і той же день, прийнятна точність розпізнавання, як правило, становить 95%.

Для зображень, зроблених різними апаратами і при різному освітленні, точність, як правило, падає до 80%. Для зображень, зроблених з різницею в рік, точність розпізнавання склало приблизно 50%. При цьому варто зауважити, що навіть 50 відсотків - це більш ніж прийнятна точність роботи системи подібного роду.

Щорічно FERET публікує звіт про порівняльному випробуванні сучасних систем розпізнавання обличчя на базі обличчя більше одного мільйона. На превеликий жаль в останніх звітах не розкриваються принципи побудови систем розпізнавання, а публікуються тільки результати роботи комерційних систем. На сьогоднішній день лідируючої є система NeoFace розроблена компанією NEC.

1.3. Архітектура нейронних мереж

Більшість нейронних мереж складаються з формальних нейронів.[14]

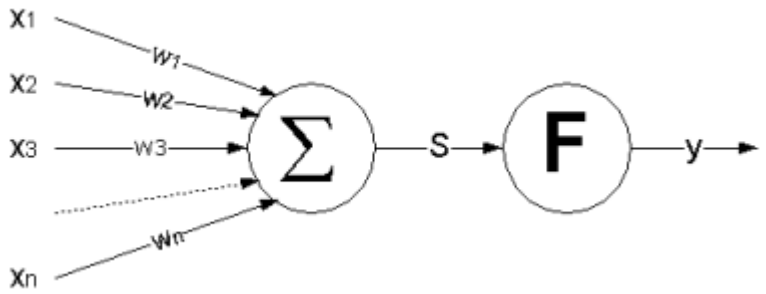


Рис 1.9. Формальний нейрон.

Тут x1..xn - значення, що надходять на входи (синапси) нейрона, w1..wn - ваги синапсів (можуть бути як гальмуючі, так і підсилюючі), S - зважена сума вхідних сигналів.

$$S = \sum_{i=1}^{n} W_i\, X_i - T = (W, X) - T = |W| * |X| * \cos \propto -T,$$

де T - поріг нейрона (у багатьох моделях обходяться без нього), F - функція активації нейрона, що перетворює зважену суму в вихідний сигнал. Зважена сума вхідних сигналів може бути інтерпретована як проекція вхідного вектора на вектор ваг, де α - кут між цими векторами.

Формальний нейрон моделює деякі властивості біологічного нейрона. Набір пов'язаних формальних нейронів являє собою штучну нейронну мережу. Штучні нейронні мережі здатні виконувати будь-які логічні операції і взагалі будь-які перетворення, реалізовані дискретними пристроями з кінцевої пам'яттю (інше питання в тому, як налаштувати ваги такої мережі). Нейрони в природних нейронних мережах набагато складніші, їх функціонування є

складним процесом, протяжним в часі. Крім того, існує думка, що мозок має квантової структурою, а процес мислення заснований на квантових ефектах. Мозок людини складається з 10 трильйонів нейронів, пов'язаних між собою 1014 синапсами. Такі обчислювальні потужності сучасної обчислювальної техніки поки недоступні. Структура мозку визначена генетично від народження, а зв'язку між нейронами розвиваються і модифікуються протягом усього життя, тобто свій інтелектуальний досвід людина отримує в процесі навчання. Це говорить про перспективність розвитку штучних нейронних мереж.

За характером зв'язків нейронні мережі можуть бути повнозв'язні, коли кожен нейрон пов'язаний з усіма іншими, і шаруватими, коли нейрони наступного шару пов'язані тільки з усіма нейронами попереднього шару. Ці дві архітектури є базовими, але можливі і різні варіації. За характером функціонування нейронні мережі можуть бути однопрохідними, коли вихід мережі розраховується за один прохід мережі, і релаксаційним, коли функціонування мережі триває до досягнення стабільного стану, це стан і є результатом роботи.

За характером формування зв'язків нейронні мережі можуть бути наступних видів: [14]

- Навчання з учителем. Зв'язки налаштовуються в процесі навчання, причому еталонні значення результатів роботи відомі.

- Самонавчання (навчання без вчителя). еталонні результати невідомі (не потрібні), мережа в процесі навчання повинна організувати вхідні образи на основі їх подібності.

- Фіксований зв'язок. Визначаються характером розв'язуваної задачі (наприклад, в оптимізаційних задачах).

Нейронні мережі можуть також відрізнятися типом вхідної інформації (двійкова, аналогова і т.п.) і методом навчання.

1.3.1. Згорткова нейронна мережа

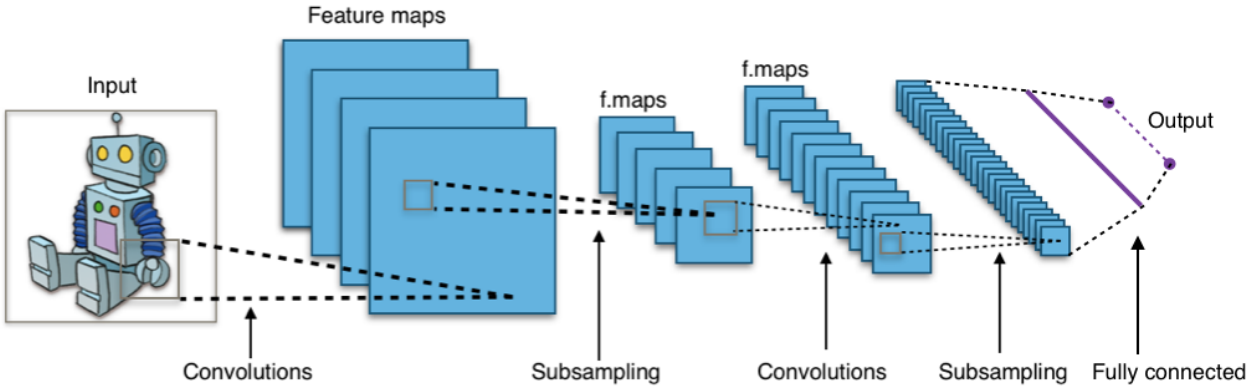


Рис 1.10. Типова архітектура загорткової нейронної мережі

У звичайному перцептроні, який представляє собою повнозв'язну нейронну мережу, кожен нейрон пов'язаний з усіма нейронами попереднього шару, причому кожний зв'язок має свій персональний ваговий коефіцієнт. У згортковій нейронної мережі в операції згортки використовується лише обмежена матриця ваг невеликого розміру, яку «рухають» по всьому оброблюваного шару (на самому початку - безпосередньо по вхідному зображенню), формуючи після кожного зсуву сигнал активації для нейрона наступного шару з аналогічною позицією. Тобто для різних нейронів вихідного шару використовуються одна і та ж матриця ваг, яку також називають ядром згортки. Її інтерпретують як графічне кодування якої-небудь ознаки, наприклад, наявність похилої лінії під певним кутом. Тоді наступний шар, що вийшов в результаті операції згортки такою матрицею ваг, показує наявність даної ознаки в оброблюваному шарі і її координати, формуючи так звану карту ознак (англ. Feature map).

Природно, в згортковій нейронної мережі набір ваг не один, а ціла гама, що кодує елементи зображення (наприклад лінії і дуги під різними кутами). При цьому такі ядра згортки не закладаються дослідником заздалегідь, а формуються самостійно шляхом навчання мережі класичним методом зворотного поширення помилки. Прохід кожним набором ваг формує свій власний примірник карти ознак, роблячи нейронну мережу багатоканальної

(багато незалежних карт ознак на одному шарі). Також слід зазначити, що при переборі шару матрицею ваг її пересувають зазвичай не на повний крок (розмір цієї матриці), а на невелику відстань. Так, наприклад, при розмірності матриці ваг 5 × 5 її зрушують на один або два нейрона (пікселя) замість п'яти, щоб не «переступити» шукану ознаку.

Операція Субдіскретізація (англ. Subsampling, англ. Pooling, також перекладається як «операція підвибірки» або операція об'єднання), виконує зменшення розмірності сформованих карт ознак. У даній архітектурі мережі вважається, що інформація про факт наявності шуканого ознаки важливіше точного знання його координат, тому з кількох сусідніх нейронів карти ознак вибирається максимальний і приймається за один нейрон ущільненої карти ознак меншої розмірності. За рахунок цієї операції, крім прискорення подальших обчислень, мережа стає більш інваріантної до масштабу вхідного зображення.

Мережа складається з великої кількості шарів. Після початкового шару (вхідного зображення) сигнал проходить серію згорткових шарів, в яких чергується власне згортка і Субдіскретізація (пулінг). Чергування шарів дозволяє складати «карти ознак» з карт ознак, на кожному наступному шарі карта зменшується в розмірі, але збільшується кількість каналів. На практиці це означає здатність розпізнавання складних ієрархій ознак. Зазвичай після проходження декількох шарів карта ознак вироджується в вектор або навіть скаляр, але таких карт ознак стають сотні. На виході згорткових шарів мережі додатково встановлюють кілька шарів повно нейронної мережі (перцептрон), на вхід якого подаються кінцеві карти ознак.

1.4. Висновки до першого розділу

В наш час системи розпізнавання осіб є невід'ємною частиною життя людини.

Технології розпізнавання осіб застосовуються в найрізноманітніших сферах:

-забезпечення безпеки в місцях великого скупчення людей;

-системи охорони, уникнути незаконного проникнення на територію об'єкта, пошук зловмисників;

-фейс-контроль в сегменті громадського харчування та розваг, пошук підозрілих і потенційно небезпечних відвідувачів;

-верифікація банківських карт;

-онлайн-платежі;

-контекстна реклама, цифровий маркетинг, Intelligent Signage і Digital Signage;

-фототехніка;

-криміналістика;

-телеконференції;

-мобільні додатки;

-пошук фото у великих базах фотознімків;

На даний момент системи розпізнавання обличчя мають обширні можливості. Наприклад, соціальна мережа Facebook навчилась розпізнавати ваших друзів на ваших фотографіях. У старі часи Фейсбук відзначав ваших друзів на фотографіях лише після того, як ви обирали відповідне зображення і вводили через клавіатуру ім'я вашого друга. Зараз одразу після завантаження фотографії Фейсбук відзначає сторінки ваших друзів автоматично за допомогою розпізнавання обличчя.

Серед існуючих засобів роспізнавання обличчя можна виділити основні: метод порівняння на графах, нейронні мережі, метод головних компонент, метод Віоли-Джонса. Серед них найбільш ефективними є нейронні мережі, однак і вони мають низку недоліків: проблема освітленості, повороту голови. Незважаючи на це, саме метод з нейронними мережами був обраний для розробки програми по розпізнаванню обличчя у режимі реального часу.

РОЗДІЛ 2
РОЗРОБКА СИСТЕМИ РОЗПІЗНАВАННЯ ОБЛИЧЧЯ

2.1. Середовище та платформа розробки

Дана програма буде реалізована на основі веб інтерфейсу в форматі html/php з використанням коду на мові JavaScript з підключенням бібліотеки Faceapi. При створенні системи був використований редактор коду Visual Code, тому що це досить популярний редактор, який має свої переваги:

Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціонується як легковагове рішення, що дозволяє обійтися без повного інтегрованого середовища розробки. Серед підтримуваних мов і технологій:

JavaScript, C++, C#, TypeScript, jade, PHP, Python, XML, Batch, F#, DockerFile, Coffee Script, Java, HandleBars, R, Objective-C, PowerShell, Luna, Visual Basic, Markdown, JSON, HTML, CSS, LESS і SASS, Haxe та і найголовніше, цей редактор безкоштовний.

Інтерфейс редактора.

Інтерфейс Visual Code розділений на п'ять основних областей, які можна легко налаштувати (Рис. 2.2):

1) Activity Bar: дозволяє перемикатися між уявленнями: провідник, пошук, контроль версій, налагодження та розширення.

2) Side Bar: містить активний вид.

3) Editor: тут можна редагувати файли і переглядати файли уцінки. Є можливість розстановки кілька відкритих файлів поруч.

4) Panel: відображає різні панелі: вбудований термінал, панелі виводу для налагоджувальної інформації, помилок і попереджень.

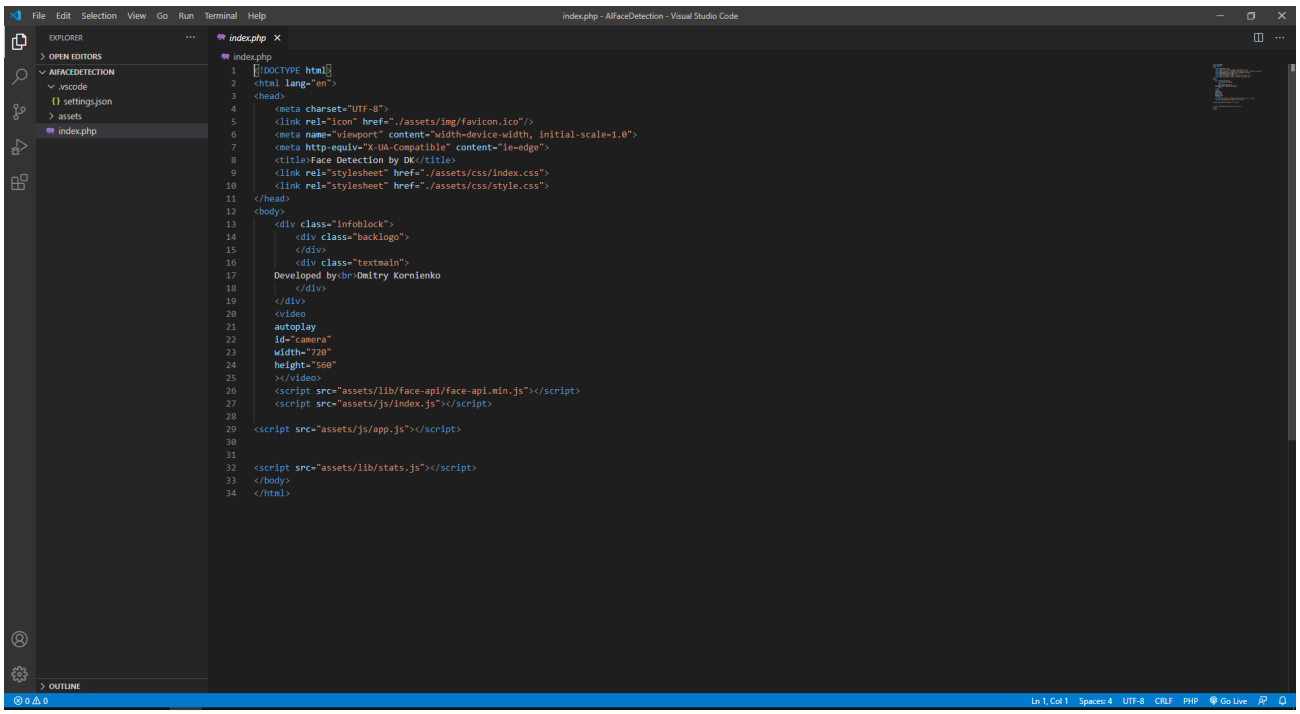


Рис 2.1. Інтерфейс редактора Visual Code

5) Status: відображає інформацію про поточний відкритому проекті і файлі. Також містить кнопки для виконання дій з управління версіями, а також для включення / відключення функцій розширення.

Також є верхній рядок меню, де є можливість отримати доступ до системи меню редактора. Для користувачів Linux вбудованим терміналом за замовчуванням, ймовірно, буде оболонка Bash. Для користувачів Windows це PowerShell. На щастя, всередині списку терміналу знаходиться селектор оболонки, який дозволить вибрати іншу оболонку. Якщо встановлено, ви можете вибрати будь-який з наступних:

- Command Prompt
- PowerShell
- PowerShell Core
- Git Bash
- WSL Bash

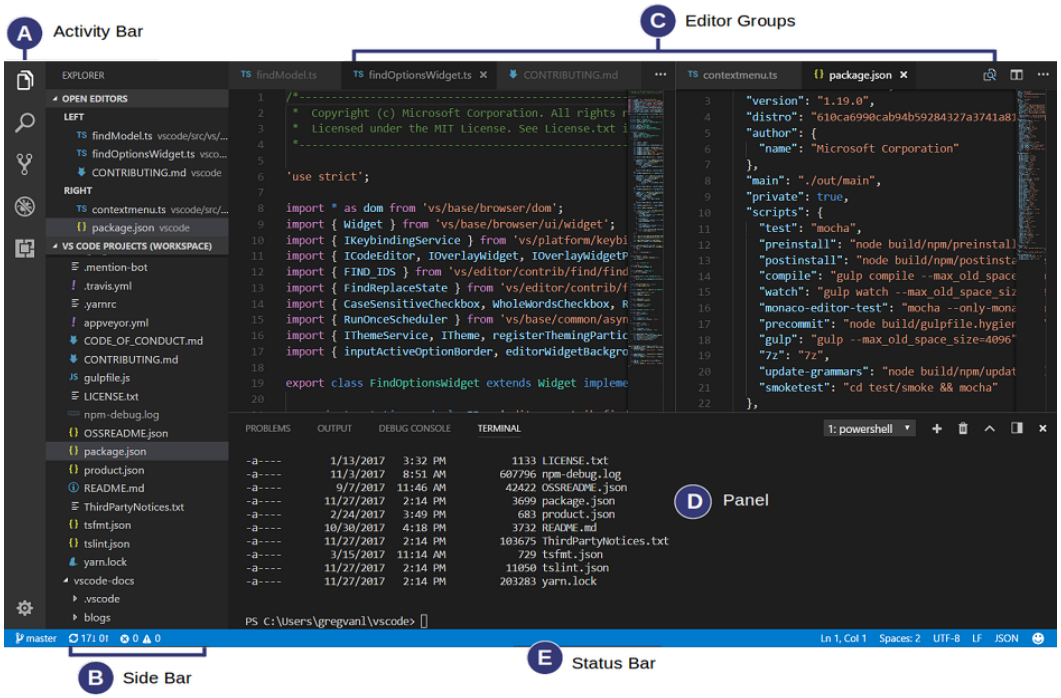


Рис 2.2. Основні області редактора

Для запуску программи на комп’ютері необхідний локальний сервер. Був обраний Open Server Panel - це портативна серверна платформа і програмне середовище, створене спеціально для веб-розробників.

Програмний комплекс має обширний набір серверного програмного забезпечення, а також можливості з адміністрування та налаштування компонентів. Платформа широко використовується з метою розробки, налагодження і тестування веб-проектів і для надання веб-сервісів в локальних мережах.

Хоча спочатку програмні продукти, що входять до складу комплексу, не розроблялись спеціально для роботи один з одним, така зв'язка стала дуже популярною серед користувачів Windows, в першу чергу через те, що вони отримували безкоштовний комплекс програм з надійністю на рівні Linux серверів.

2.2. Загальна структура проекту

Структура проекту наступна:

1) Створення папки с проектом з назвою AIFaceDetection.
2) Створення index.php файлу.
3) Далі в цій папці створення папки assets.
4) В папці assets буде ще три папки:

- css;
- js;
- lib.

5) В папці css буде файл index.css для налаштування стилів.
6) В папці js знаходиться файл index.js.

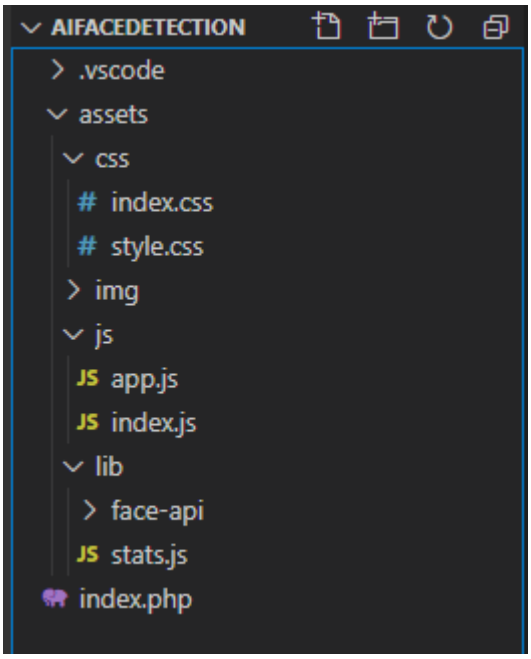


Рис 2.3. Структура проекту

У файлі index.php прописуються основна інформація щодо сайту, а також підключення скриптів js. Вони підключаються за допомогою команди <script src="assets/js/index.js"></script>. Також був доданий тег body: <video autoplay id="camera" width="720" height="560"></video> для доступу до відео.

index.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="icon" href="./assets/img/favicon.ico"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Face Detection by DK</title>
    <link rel="stylesheet" href="./assets/css/index.css">
    <link rel="stylesheet" href="./assets/css/style.css">
</head>
<body>
    <div class="infoblock">
        <div class="backlogo">
        </div>
        <div class="textmain">
    Developed by<br>Dmitry Kornienko
        </div>
    </div>
    <video
    autoplay
    id="camera"
    width="720"
    height="560"
    ></video>
    <script src="assets/lib/face-api/face-api.min.js"></script>
    <script src="assets/js/index.js"></script>

<script src="assets/js/app.js"></script>


<script src="assets/lib/stats.js"></script>
</body>
</html>
```

Рис 2.4. Вміст файлу index.php

2.3. Підключення відеопристрою

Наступним кроком буде редагування файлу index.js, який знаходиться в папці js:

Виконується доступ до медіапристроїв navigator.mediaDevices.enumerateDevices(), далі йде перебір всіх знайдених медіапристроїв: if (Array.isArray(devices)) { devices.forEach(device => { if (device.kind === "videoinput") {,

передаються дані камери та запрошується доступ у користувача на використання відеопристрою: if (device.label.includes("")) {

navigator.getUserMedia({

video: {

```
deviceId: device.deviceId,
                  },
                },
                (stream) => (camera.srcObject = stream),
                (error) => console.error(error)
              );
            }
```

Далі транслюються відеодані с камери:

```
(stream) => (camera.srcObject = stream),
                    (error) => console.error(error).
```

2.4. Підключення моделей розпізнавання обличчя. Файл index.js

Далі у файлі додається на початку кода змінна const camera = document.getElementById("camera")

Після цього у папці lib, яка знаходиться в папці assets буде знаходитись файл face-api.min.js та папка face-api, в якій знаходяться папка labels, де знаходяться фотографії людей, яких потрібно розпізнати, та папка models, в якій знаходяться деякі функції для розпізнавання, наприклад емоції, стать, вік, та згорткова нейронна мережа SSD MobilNet V1.

У файлі index.js додаються моделі, які знаходяться в папці lib/models, також додається архітектура згорткової нейронної мережі SSD MobilNet V1.

return Promise.all(labels.map(async label =>

Метод Promise.all () повертає обіцянку, яку виконує тоді, коли будуть виконані всі обіцянки, передані у вигляді перераховуваного аргументу, або відхилено будь-яке з переданих обіцянок.

Метод map () створює новий масив з результатом виклику зазначеної функції для кожного елемента масиву.

```
const descriptions = []
    for (let i = 1; i <= 2; i++) {
        const img = await faceapi.fetchImage(`/assets/lib/face-api/labels/${label}/${i}.jpg`)
        const detections = await faceapi
          .detectSingleFace(img)
          .withFaceLandmarks()
          .withFaceDescriptor()
        descriptions.push(detections.descriptor)
      }
```

У цій частині коду йде підключення файлів з папки assets/lib/face-api/labels.

return new faceapi.LabeledFaceDescriptors(label, descriptions) // повертає новий опис по мітці

Далі йде підключення моделей розпізнавання з бібліотеки Faceapi. Підключаються такі моделі як: захват обличчя, розпізнавання по точкам, розпізнавання по імені, по емоціям, по статі, SSD - детектор обличчя, Mobilenetv1 - архітектура згорткової нейронної мережі.

```
Promise.all([
    faceapi.nets.tinyFaceDetector.loadFromUri('/assets/lib/face-api/models'),
    faceapi.nets.faceLandmark68Net.loadFromUri('/assets/lib/face-api/models'),
    faceapi.nets.faceRecognitionNet.loadFromUri('/assets/lib/face-api/models'),
    faceapi.nets.faceExpressionNet.loadFromUri('/assets/lib/face-api/models'),
    faceapi.nets.ageGenderNet.loadFromUri('/assets/lib/face-api/models'),
    faceapi.nets.ssdMobilenetv1.loadFromUri('/assets/lib/face-api/models'),
]).then(startCamera)
```

Рис 2.5. Підключення моделей розпізнавання з бібліотеки Faceapi

```
.detectAllFaces(
        camera,
        new faceapi.TinyFaceDetectorOptions()
```

Метод detectAllFaces виявить всі особи в відео і приймає в якості параметрів елемент відео і модель примірника TinyFaceDetectorOptions. Це потрібно, щоб адаптувати ці виявлення до розміру камери на якому зосереджені, тому що він повинен знати, що обличчя тут, в цій позиції, щоб він міг намалювати квадрат і поставити:

.withFaceLandmarks()

.withFaceExpressions()

.withAgeAndGender()

.withFaceDescriptors()

Метод detectAllFaces виявить всі обличчя в відео і приймає в якості параметрів елемент відео і модель примірника TinyFaceDetectorOptions. Метод withFaceLandmarks виявляє такі частини людського обличчя, як ніс, губи, очі, лінії щелепи і т. Д. Метод withFaceLandmarks виявляє вираз обличчя, таке як щасливе, нейтральне, зле і т. Д.

Метод resizeResults обчислює розмір виявлених осіб відповідно до розміру екрана відео і точно відповідає виявленому обличчю. В якості параметра він буде приймати displaySize, який являє собою висоту і ширину відеоелемента.

getContext встановлює контекст для відображення виявлення в форматі 2d або 3d. Перед установкою параметра after, методcontext clearRect очистить попередні виявлення.

drawDetections буде малювати прямокутник на виявленому обличчі і приймати в якості параметра полотно і resizedDetections. resizedDetections отримує нові обчислення від методу resizeResults, який приймає значення виявлення і displaySize як параметр для генерації результату.

Метод drawFaceLandmarks виділить частини особи, такі як ніс, губи, очі і т. Д., І приймає як параметр полотно і resizedDetections.

Метод drawFaceExpressions визначає вираз обличчя і приймає як параметр полотно і resizedDetections.

Потрібно запросити отримання креслення для виявлення, і щоб була інформація.

const resizedDetections = faceapi.resizeResults(detections, canvasSize)

Canvas - елемент HTML5, призначений для створення двовимірного зображення за допомогою скриптів, зазвичай на мові JavaScript.

canvas.getContext ('2d'). clearRect (0, 0, canvas.width, canvas.height)

Якщо виявлення з правильним розміром, то зберігається відео: faceapi.draw.drawDetections (canvas, resizedDetections).

results.forEach ((result, index) => Метод forEach () виконує зазначену функцію один раз для кожного елемента в масиві.

const box = resizedDetections [index] .detection.box - малюється новий прямокутник, поки не з'явиться текст

const {label, distance} = result - результат, що буде на екрані.

Далі додається модель розпізнавання обличчя через 68 крапок, утворюючи форму **Active Appearance Models (AAM).** Активні моделі зовнішнього вигляду (Active Appearance Models, AAM) - це статистичні моделі зображень, які шляхом різного роду деформацій можуть бути підігнані під реальне зображення. Даний тип моделей в двовимірному варіанті було запропоновано Тімом Кутса і Крісом Тейлором в 1998 році. Спочатку активні моделі зовнішнього вигляду застосовувалися для оцінки зображень обличчя.

Активна модель зовнішнього вигляду містить два типи параметрів: параметри, пов'язані зі статистичною моделлю пікселів зображення або текстурою (параметри зовнішнього вигляду). Перед використанням модель повинна бути навчена на безлічі заздалегідь розмічених зображень. Розмітка зображення вручну. Кожна мітка має свій номер і визначає конкретну точку, яка повинна знайти модель під час адаптації до нового зображення.

2.5. Розпізнавання конкретної особи. Результат роботи програми

В даній програмі присутня функція розпізнавання обличчя конкретної

людини за допомогою фотографії обличчя, яка зберігається у певній папці.

Рис 2.6. Директорія з фотографіями для розпізнавання

Розпізнавання відбувається за допомогою наступного коду:

```
const labels = ['Andrew', 'Dmitry']
    return Promise.all(labels.map(async label
      const descriptions = []
       for (let i = 1; i <= 2; i++) {
          const img = await faceapi.fetchImage(`/assets/lib/face-
api/labels/${label}/${i}.jpg`)
```

Результат розпізнавання та роботи програми в цілому можна спостерігати на рисунку наведеному нижче.

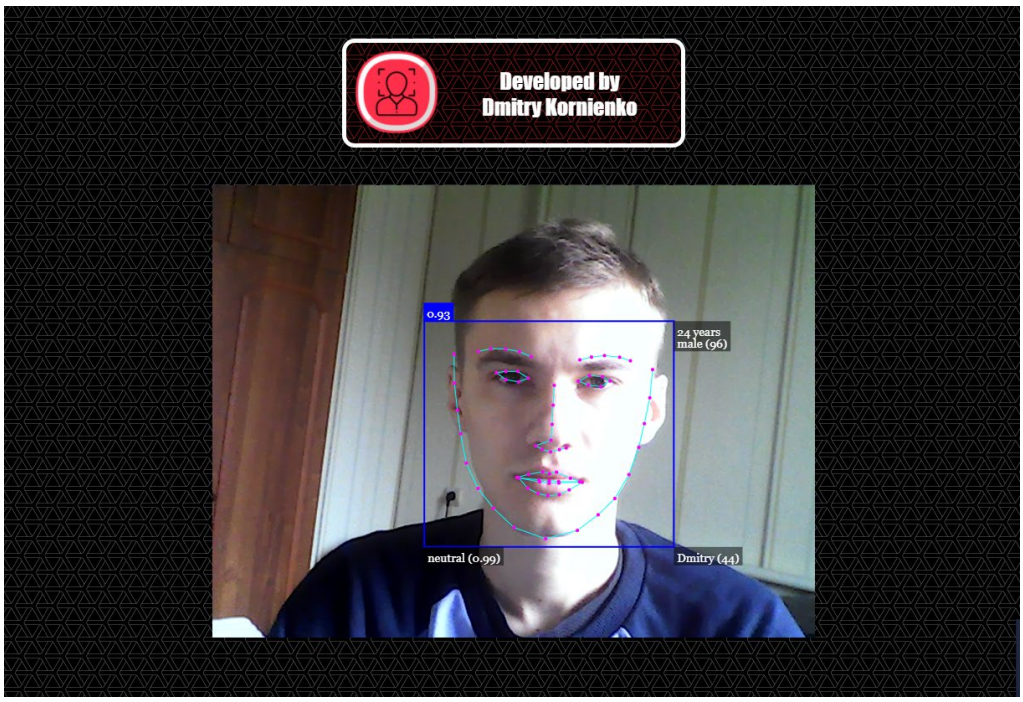


Рис 2.7. Результат розпізнавання обличчя

Зліва зверху позначається ймовірність того, що обличчя було розпізнане (у даному випадку це 93%).

У правому верхньому кутку визначається приблизний вік людини яка була розпізнана, а також стать (96% у даному випадку).

Зліва снизу визначається емоційний стан обличчя на основі положення ліній рота, брів, очей. На зображенні цей стан нейтральний(99%).

У правому нижньому кутку визначається конкретна людина із існуючих данних які знаходяться у папці labels в директорії assets.

Тепер потрібно перевірити наскільки добре програма розпізнає обличчя, якщо, прикрита його частина (досить актуально в 2020).

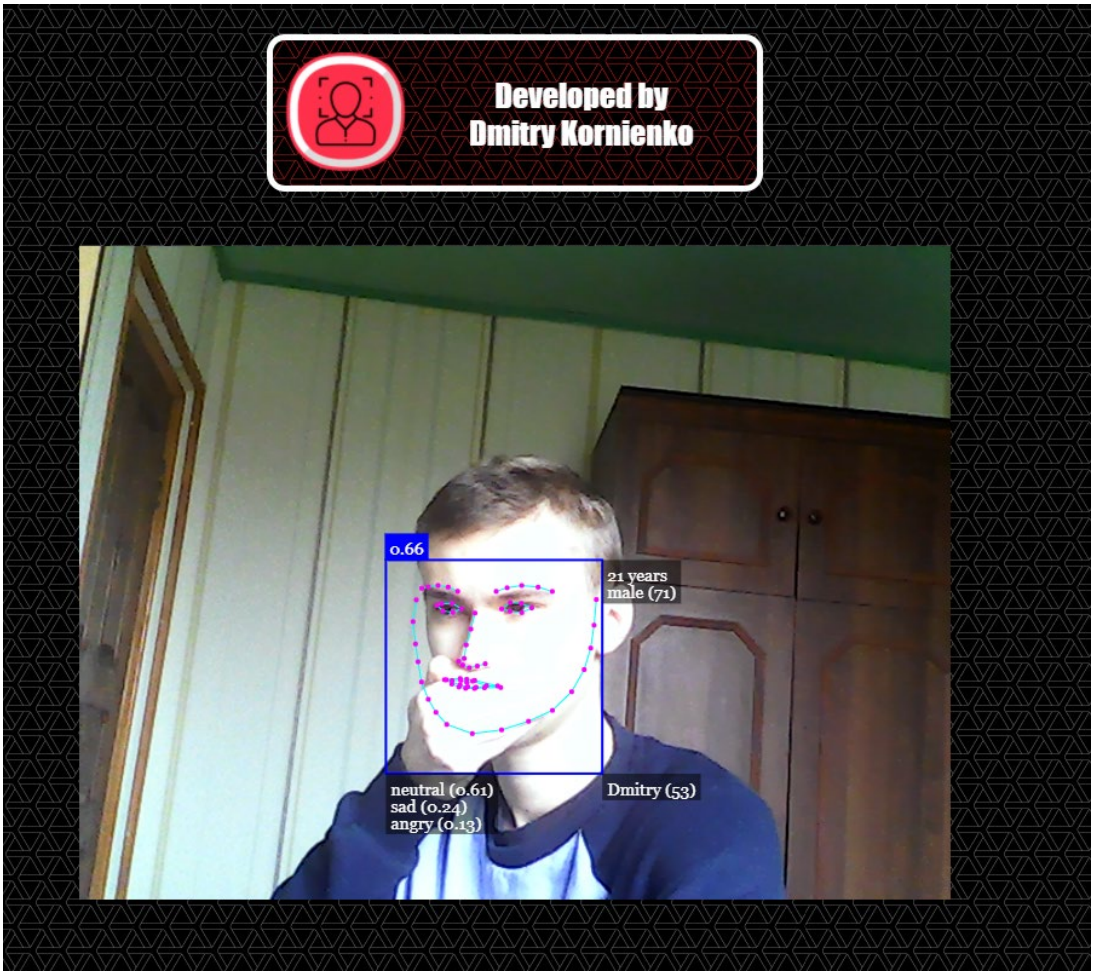


Рис 2.8. Розпізнавання частково прикритого обличчя

Як можна бачити, програма без проблем розпізнає обличчя з прикритим

ротом, і навіть розпізнає конкретну людину. Але що буде якщо ускладнити процедуру розпізнавання ще більше?

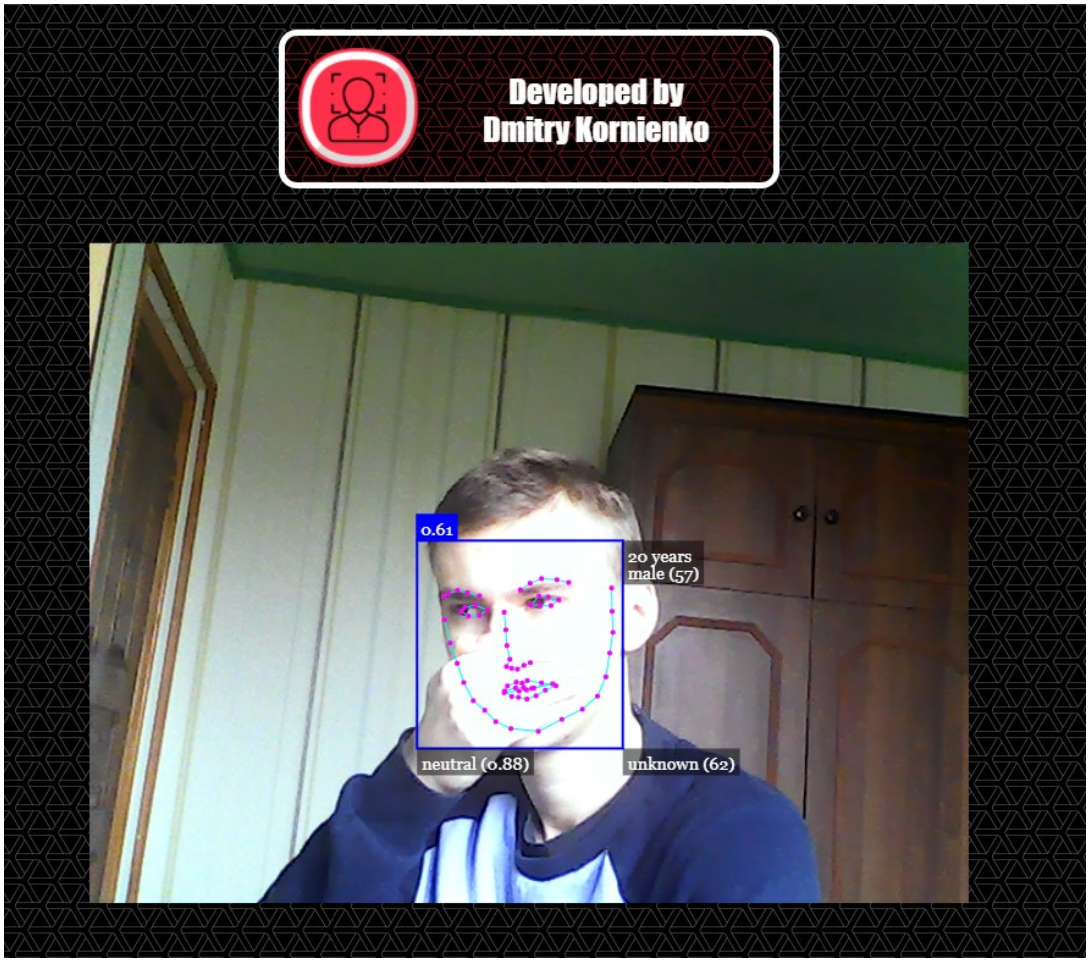


Рис 2.9. Розпізнавання майже повністю прикритого обличчя

Як можна бачити на цьому зображенні, програма має проблеми з розпізнаванням якщо прикрити рот і ніс. Обличчя буде розпізнане, проте конкретну особу з бази даних розпізнати не вдалось. Варто зазначити що тести проводились з певними проблемами в освітленні при денному світлі (надто сильне освітлення з боку).

2.6. Висновки до другого розділу

В цьому розділі була розроблена програма на мові JavaScript та середовищі розробки Visual Code, також була взята архітектура згорткової нейронної мережі MobilNet.

Дана програма здатна розпізнавати обличчя у режимі реального часу з вебкамери або відеокамери. За допомогою її інтеграції у браузер є можливість віддаленого доступу та інші особливості які забезпечують зручність використання. Також є функція розпізнання певної особи по фотографії, що дає додаткові можливості.

Програма непогано проявила себе у тестах зі спробами затруднити розпізнавання. Був прикритий спочатку рот, потім рот з носом. В результаті в першому випадку програма впоралась з розпізнаванням, а в другому випадку не вдалось розпізнати конкретну особу але обличчя все ж було розпізнане.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ СИСТЕМИ

3.1. Перелік експериментів і система на OpenCV

В попередньому розділі була розроблена комп’ютерна система для розпізнавання обличчя на основі згорткової нейронної мережі. В цьому розділі буде порівняно розроблену систему з існуючими на JavaScript, через те що ця система розроблена на цій мові програмування.

Розроблена система, яка побудована на архітектурі Tensorflow буде порівняна з системою яка побудована на бібліотеці комп’ютерного зору OpenCV. OpenCV (Open Source Computer Vision Library, бібліотека комп'ютерного зору з відкритим вихідним кодом) - бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. Реалізована на C / C ++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов[15]. Може вільно використовуватися в академічних і комерційних цілях - поширюється в умовах ліцензії BSD. Для того, щоб розпізнати обличчя у цій системі виділяються основні його компоненти, такі як ніс, лоб, очі, губи і т.д. Для цього використовуються шаблони (вони ж примітиви Хаара). Якщо шаблони відповідають конкретним областям на зображенні то тоді вважається, що на зображенні є людське обличчя. Насправді подібних шаблонів набагато більше. Для кожного з них вважається різниця між яскравістю білої і чорної областей. Це значення порівнюється з еталоном і приймається рішення про те, чи є тут частина людського обличчя чи ні.

В даній програмі також присутня функція розпізнавання конкретної особи. Це відбувається через браузер, після розпізнавання обличчя натискається кнопка «Add person», вписується ім’я особи, і в подальшому це обличчя буде розпізнаватись як вказане ім’я.

Експерименти будуть проходити в декількох етапах:

1) Близька дистанція;
2) Дальня дистанція;
3) Обличчя знаходиться під кутом;
4) Експеримент при поганому освітленні;

В кінці експериментів буде зроблена таблиця результатів, де будуть вписані системи та експерименти.

3.2. Експеримент на близькій дистанції

Спочатку йде розпізнавання обличчя, яке побудоване на бібліотеці OpenCV.

Рис 3.1. Результат системи, яка побудована на OpenCV – Близька дистанція

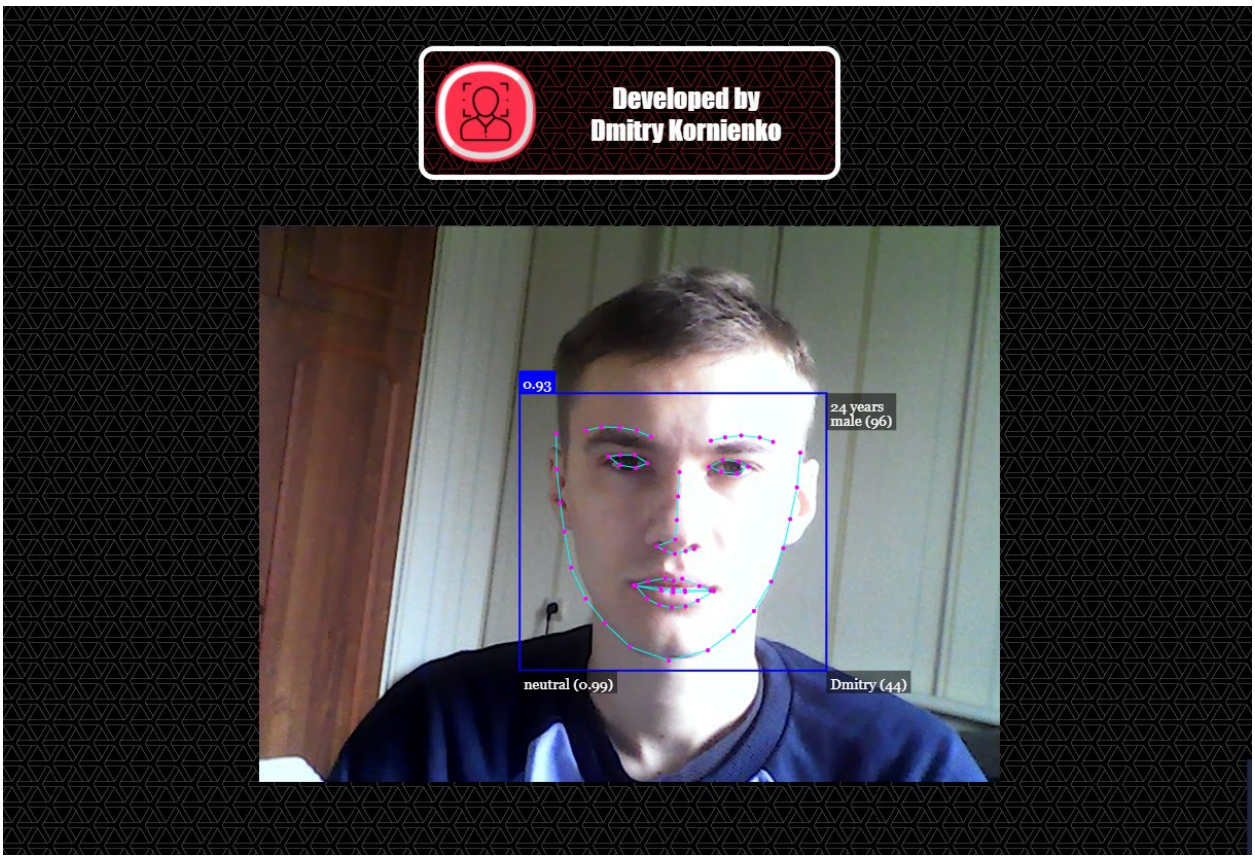


Рис 3.2. Результат розробленої системи – Близька дистанція

Як можна бачити, обидві системи впоралися з базовою задачею без проблем.

3.3. Експеримент на дальній дистанції

На цей раз завдання було ускладнено: експеримент був проведений на дальній дистанції.

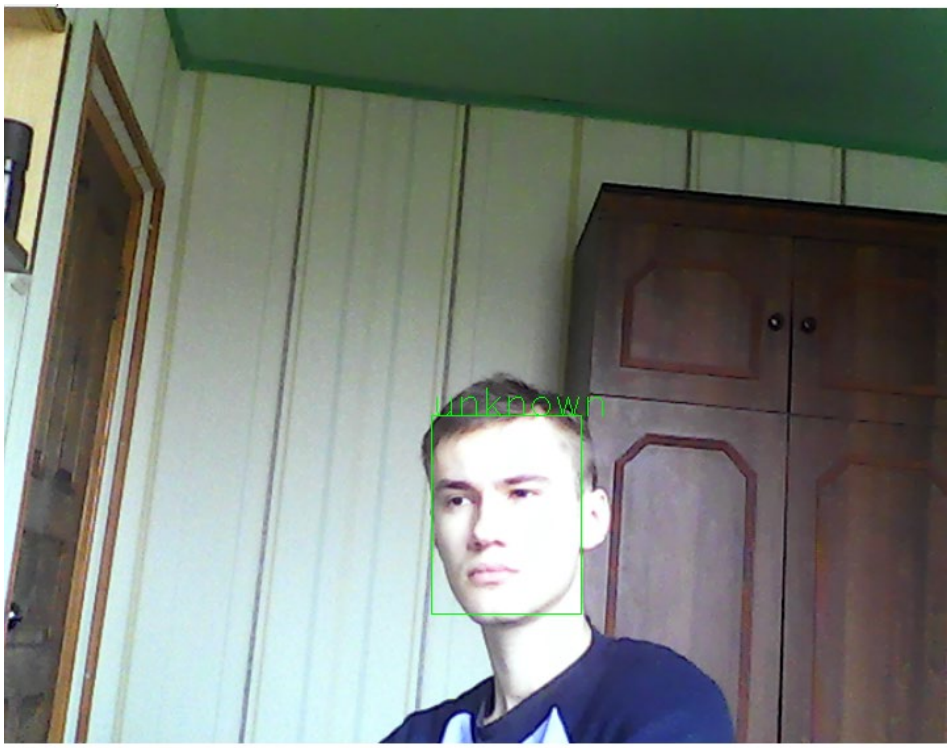


Рис 3.3. Результат системи, яка побудована на OpenCV – Дальня дистанція

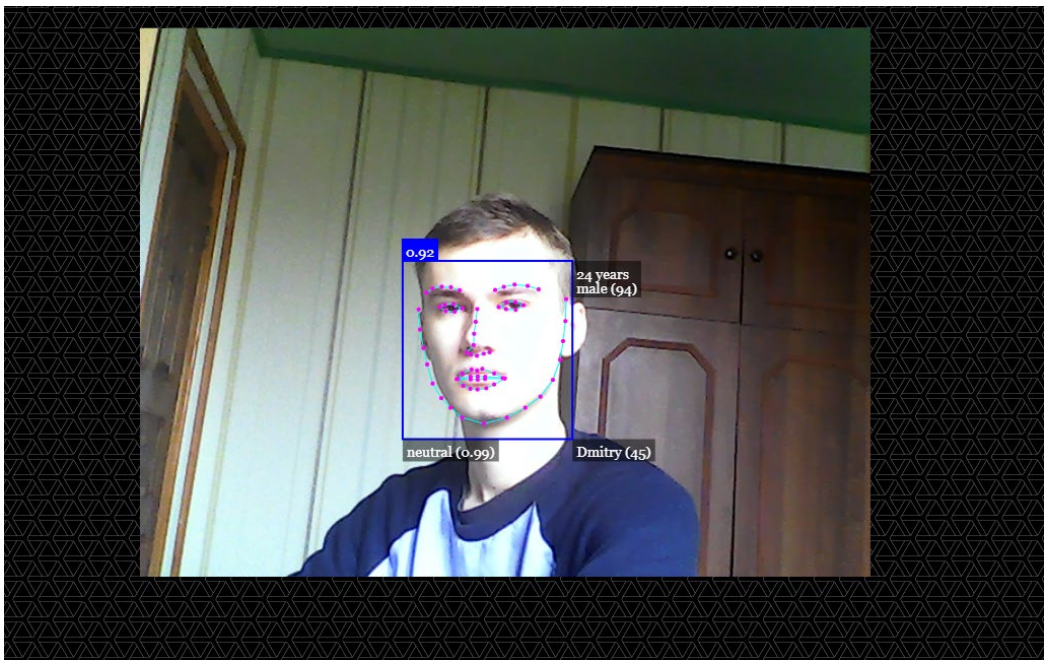


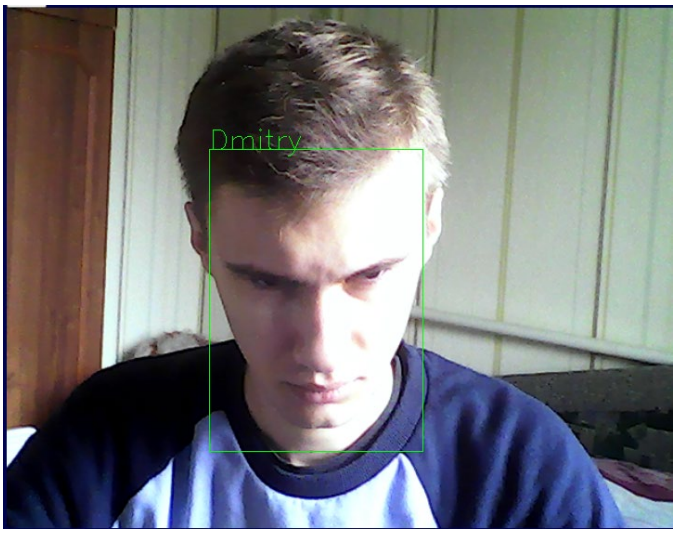
Рис 3.4. Результат розробленої системи – Дальня дистанція

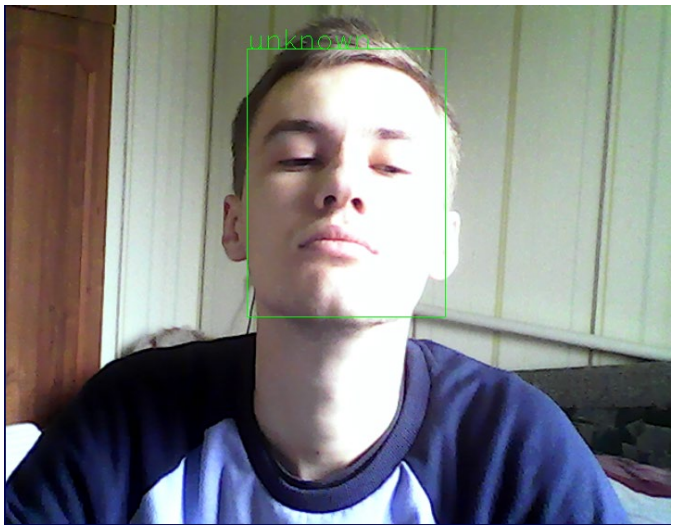
На цей раз система на OpenCV розпізнала обличчя, але на змогла розпізнати конкретну особу. Система з нейронними мережами повністю впоралась з завданням.

3.4. Експеримент з обличчям під кутом

Досить актуальний експеримент, адже часто виникають ситуації коли необхідно розпізнати обличчя коли людина не дивиться в камеру. Наприклад якщо відеокамера знаходиться в громадському місці.

Результат системи на OpenCV:





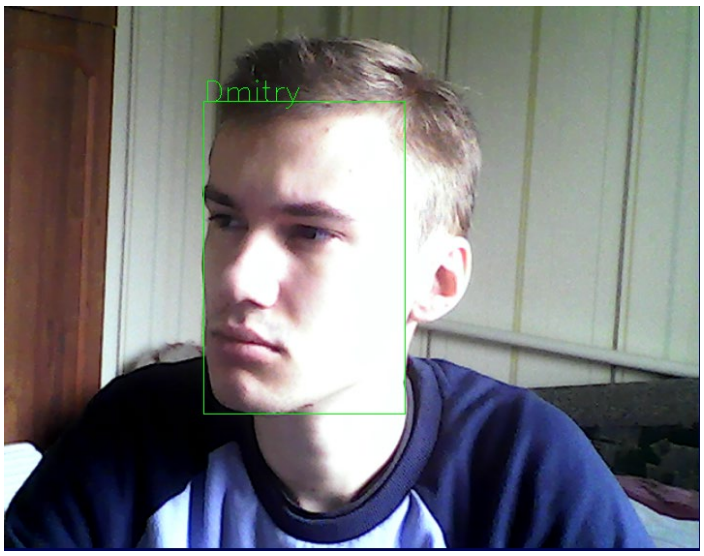
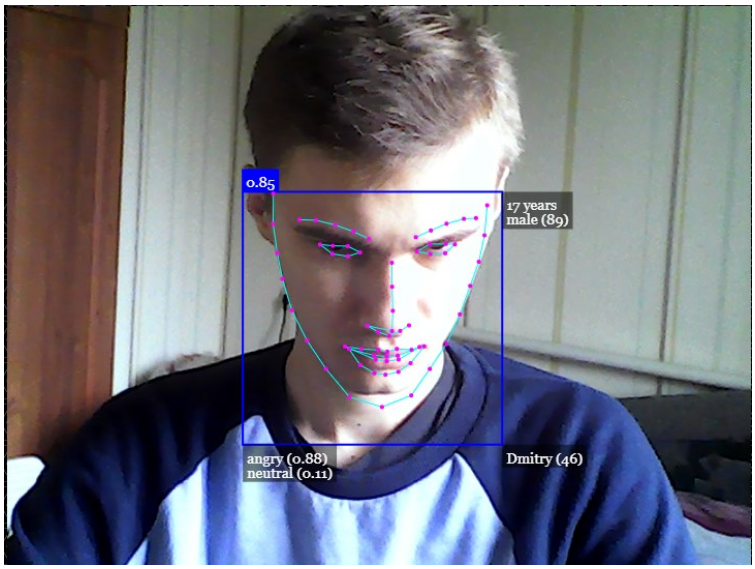
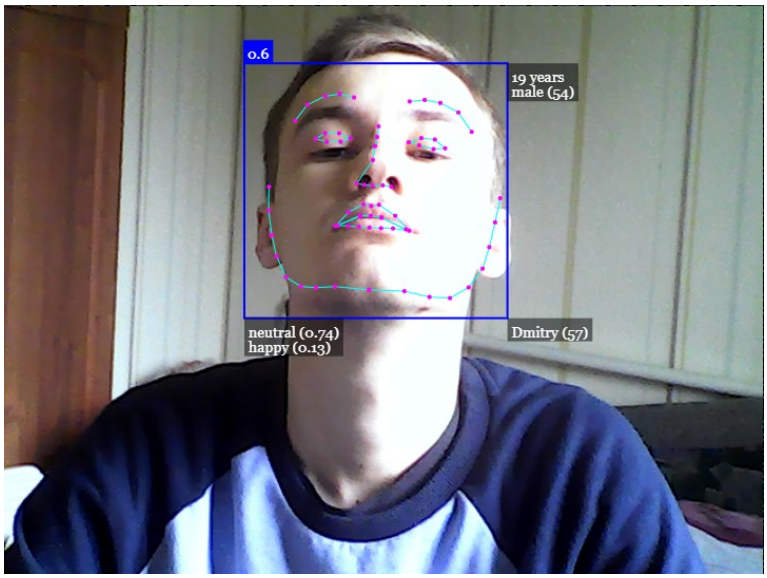


Рис 3.5. Результат системи, яка побудована на OpenCV – Обличчя під кутом

Тепер система на нейронних мережах:





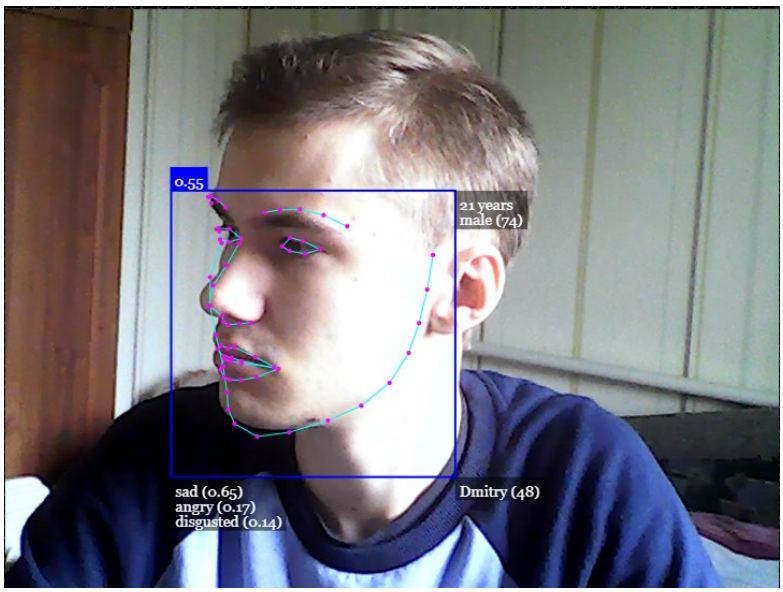


Рис 3.6. Результат розробленої системи – Обличчя під кутом

Отже, згідно з результатів рис. 3.5 та рис. 3.6, видно, що обидві системи справилися з задачею, але система на OpenCV в одному із прикладів не змогла розпізнати обличчя, тобто хто знаходиться перед нею, в той час як система з нейронними мережами виконала всі три приклада відмінно.

3.5. Експеримент при поганому освітленні

Одна з основних проблем в задачі розпізнавання обличчя пов'язана з впливом освітленості на якість розпізнавання. Більшість алгоритмів розпізнавання осіб успішно справляються зі своїм завданням, коли все зображення осіб отримані при однакових умовах освітлення. В даному експерименті буде перевірена здатність розпізнавання обличчя при поганому освітленні системами на OpenCV і розробленою системою.

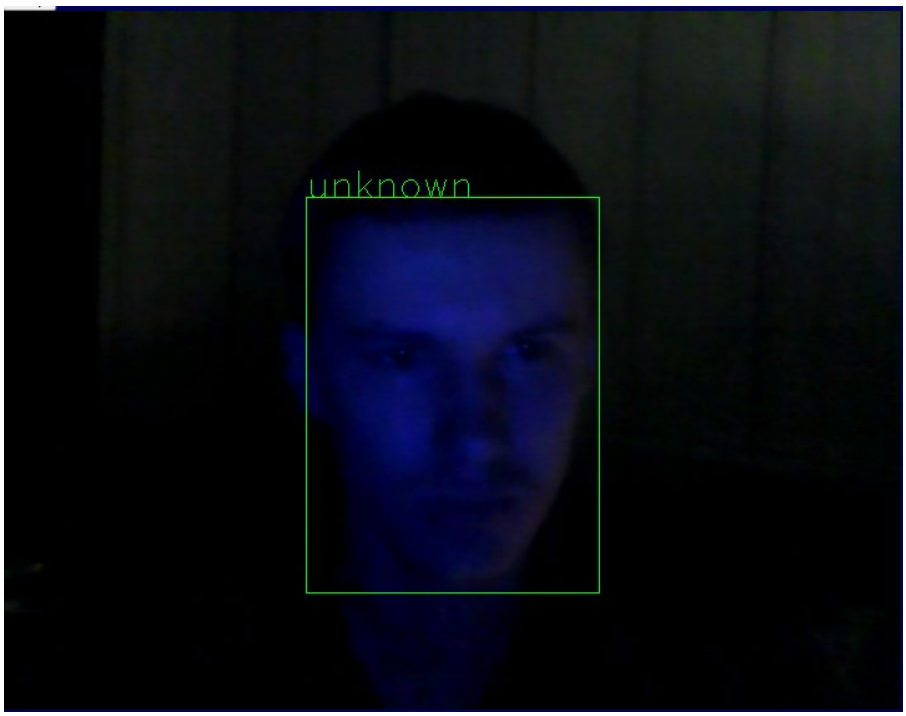


Рис 3.7. Результат системи, яка побудована на OpenCV – Погане освітлення

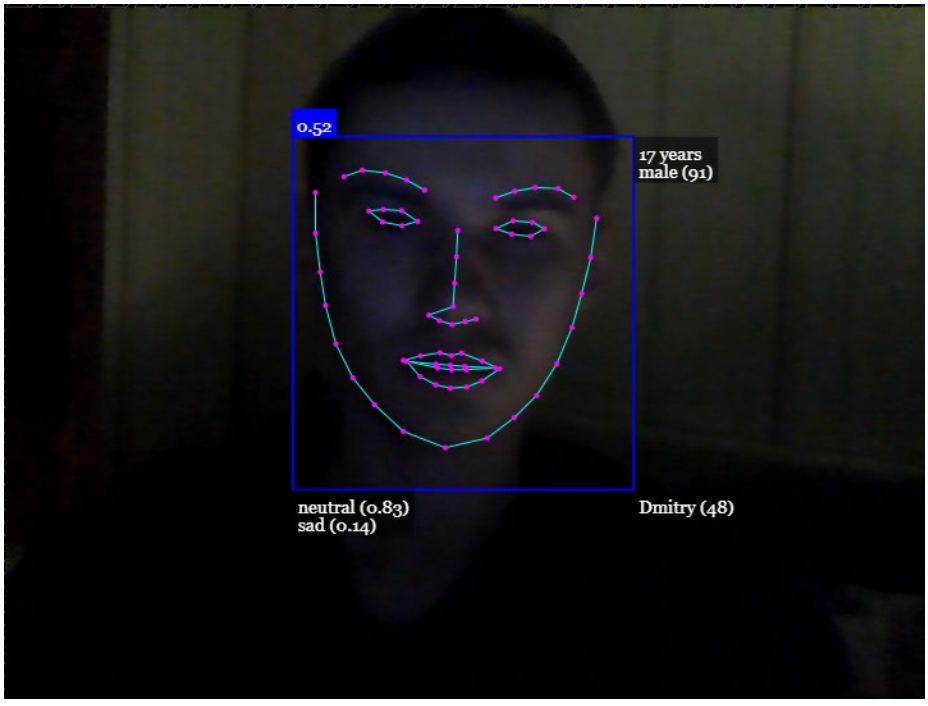


Рис 3.8. Результат розробленої системи – Погане освітлення

По експерименту в поганому освітленні, видно, що система на OpenCV та розроблена система справились з цією не простою задачею, але система на OpenCV не змогла розпізнати конкретну особу. Розроблена система експеримент виконала без проблем, але обличчя було розпізнане з ймовірністю

52%, що доказує складність експерименту.

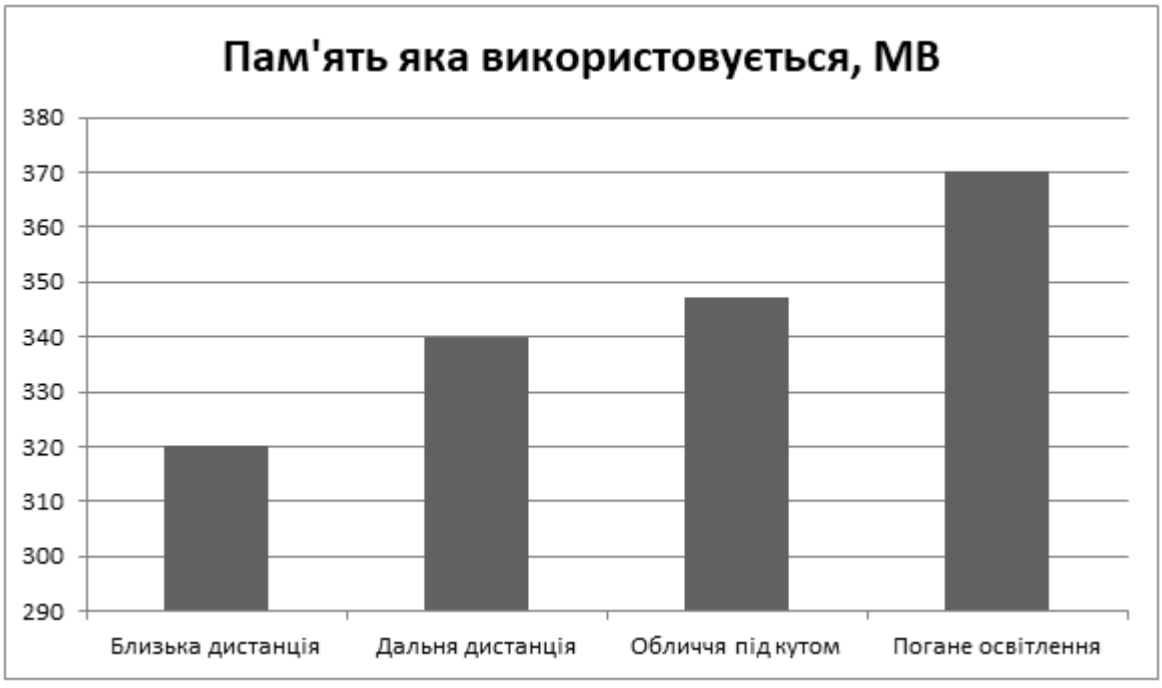
3.6. Таблиця і графіки експериментів

Таблиця 1.1. Результат експериментів

Експерименти / **Системи**	Близька дистанція	Дальня дистанція	Обличчя знаходиться під кутом	Погане освітлення
Система на OpenCV	+	+-	+-	+-
Розроблена система	+	+	+	+

Система на OpenCV добре впоралася з близькою дистанцією, в інших експериментах спостерігалися проблеми з розпізнаванням конкретної особи. Розроблена система впоралась добре в усіх експериментах.

На наступних графіках показано критерії розпізнавання (Число вірно розпізнаних зображень із загальної вибірки, час розпізнавання, пам’ять, яка використовується) та експерименти, які були при порівнянні систем.



Графік 1.1. Система на OpenCV (Пам'ять яка використовується)

Графік 1.2. Система на OpenCV (Час розпізнавання)

Графік 1.3. Розроблена система (Пам'ять яка використовується)

Графік 1.4. Розроблена система (Час розпізнавання)

По даним, які показані на графіках, можна сказати що система на OpenCV потребує більше оперативної пам’яті (320 мб проти 80 мб), а також потребує більше часу на розпізнавання.

3.7. Висновки до третього розділу

Підсумовуючи результати експериментів можна сказати, що розроблена система на нейронних мережах показала себе краще ніж система на OpenCV. В експерименті на близькій дистанції обидві системи впорались з розпізнаванням. Проте в інших експериментах система на OpenCV мала певні проблеми з розпізнаванням.

Обсяг використовуємої пам’яті був більший на системі OpenCV(приблизно 300мб проти в середньому 80 мб на розробленій системі), а час розпізнавання був також більший на цій системі. Також варто зазначити, що в системі OpenCV спостерігались значні проблеми з FPS(кількість кадрів в секунду), коли на системі з нейронними мережами такої проблеми не було.

Якщо казати про переваги системи на OpenCV, то варто зазначити, що таку систему проще розробляти, через простоту методів бібліотеки комп’ютерного зору.

РОЗДІЛ 4
ЕКОНОМІКА

4.1. Визначення трудомісткості розробки програмного забезпечення

Одним з головних етапів при розробці ПЗ є визначення трудомісткості та розрахунок витрат на створення програмного продукту. В даному розділі буде продемонстровано приклад розрахунку витрат на розробку програми з паралельними обчисленнями.

Початкові дані:

- годинна заробітна плата програміста, грн / год — 200;
- вартість машино-години ЕОМ, грн / год — 40.

Нормування роботи в процесі створення ПЗ істотно ускладнюється в силу творчого характеру роботи програміста, тому трудомісткість розробки ПЗ може бути розрахована на основі системних моделей з різною точністю оцінки.

Трудомісткість розраховується за формулою:

$$t = t_u + t_a + t_n + t_{отл} + t_д,$$

t_o – витрати праці на підготовку і опис поставленого завдання (50 год.); t_u – витрати праці на дослідження алгоритму вирішення задач;

t_a – витрати праці на розробку блок-схем алгоритму;

$t_п$ – витрати праці на програмування за готовим блок-схемою;

$t_{відл}$ – витрати праці на відладку програм на ПЕОМ;

$t_д$ – витрати праці на підготовку документації.

Сукупні витрати праці визначаються виходячи з умовного числа операторів у розроблюваному ПЗ.

Розрахунок очікуваних витрат праці:

$$Q = q \cdot C \cdot (1+ p),$$ людино-

годин, де q – передбачуване число операторів;

c – коефіцієнт складності програми;

p – коефіцієнт корекції програми в ході її розробки.

Наприклад, q буде дорівнювати 150, c – 1,4, p – 0,2.

$$Q=150 \cdot 1,4 \cdot (1+0,2) = 252$$ людино-годин,

Витрати праці на вивчення опису завдання визначаються з урахуванням уточнення опису і кваліфікації програміста за формулою:

$$t_u = \frac{\square QB}{(75...85)K'},$$

де B – коефіцієнт збільшення витрат праці (внаслідок неповного опису завдання, В= 1,2…..1,5);

K – Коефіцієнт кваліфікації програміста, який визначається в залежності від стажу роботи за фахом (якщо стаж роботи менший 2 років, то К = 1,2).

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = 252 * 1,3 / (80 * 1,2) = 3,4$$ людино-годин,

Витрати на складання програми по готовій блок-схемі:

$$t_a = \frac{\square Q}{(20...25)K}$$ людино-годин,

$t_а = 252 / 23 * 1,2 = 9,13$ людино-

годин, Витрати на складання програми по готовій

блок-схемі:

$$t_п = \frac{Q}{(20...25)K}$$ людино-годин,

$t_п = 252 /23 * 1,2 = 9,13$ людино-

годин,

Витрати на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K}$$ людино-годин,

$t_{відл} = 252 / 4 * 1,2 = 52,5$ людино-

годин, Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}$$, людино-годин,

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}$$, людино-годин,

$t_{др} = 252 / 17 * 1,2 = 12,3$ людино-годин,

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0,75 \cdot t_{др}$$, людино-годин,

$t_{до} = 0,75 * 12,3 = 9,2$ людино-годин,

$t_д = 12,3 + 9,2 = 21,5$ людино-годин,

Отримуємо трудомісткість розробки програмного забезпечення:

t = 50 +3,4 + 9,13 +9,13 + 52,5 + 21,5 = 145,6 людино-годин.

4.2. Розрахунок витрат на створення програмного забезпечення

Витрати на створення програмного забезпечення $К_{пз}$ включають витрати на заробітну плату розробників програми ($З_{зп}$) і витрати машинного часу, необхідного для налагодження програми на ЕОМ ($З_{мв}$).

$$К_{по} = З_{зп} + З_{мв} Ззп = t С_{пр}$$

де t – загальна трудомісткість розробки ПЗ;

$С_{пр}$ – середня годинна заробітна плата програміста

$$З_{зп} = 145.6 * 200 = 29120 \text{ грн.}$$

Витрати машинного часу, необхідного для налагодження програми на ЕОМ ($З_{мв}$) визначаються за формулою:

$$З_{мв} = t_{відл} \cdot С_{мч}$$

где $t_{відл}$ – трудомісткість налагодження програми на ЕОМ;

$С_{мч}$ – вартість машино-години ЕОМ.

$$З_{мв} = 52,5 \cdot 40 = 2100 \text{ грн.}$$

Таким чином витрати на створення програмного забезпечення, складуть:

$$К_{по} = 29120 + 2100 = 31220 \text{ грн.}$$

Очікувана тривалість розробки ПО:

$$T = \frac{t}{\quad}$$

$$B_k \cdot F_p$$

где $B_к$ – число розробників;

$F_р$– місячний фонд робочого часу (при 40-ка годинному робочому тижні $F_р$ = 176 годин).

T = 145,6 / 1 * 176 = 2 місяці

4.3. Маркетингові дослідження

Для розрахунку маркетингової частини на даний момент жодна з аналогічних існуючих програм з розпізнаваннм обличчя в режимі реального часу не задовольняє вимогам, тому що їх вихідний код невідомий, а тому складність розробки неможливо порівняти.

4.4 Економічна ефективність

Економічний ефект від впровадження розробленої програми очікується позитивним при невеликій вартості і складності розробки даного ПЗ, це зробить розпізнавання обличчя у різних ситуаціях більш доступним у галузі кібербезпеки та охоронній діяльності.

ВИСНОВКИ

В ході дослідження кваліфікаційної роботи було виконано розробку комп'ютерної системи розпізнавання обличчя на основі згорткової нейроннної мережі.

Для досягнення цілі потрібно було виконати наступні завдання:

- Проаналізувати існуючі підходи для розпізнавання осіб;
- Виявити класифікацію алгоритмів розпізнавання;
- Провести аналіз існуючих на ринку систем розпізнавання, виявити їхні переваги і недоліки;
- Проаналізувати основні інструментальні засоби для розробки і вибрати оптимальні з них;
- Спроектувати і програмно реалізувати систему розпізнавання.

В результаті був обраний спосіб розпізнавання за допомогою нейронних мереж і вирішені наступні задачі:

- система виявляє обличчя досить точно;
- система виявляє емоції людини;
- система виявляє рік та стать людини;
- система розпізнає саму людину;
- система працює навіть в поганих умовах.

Система розпізнавання обличчя необхідна користувачеві в першу чергу для забезпечення безпеки аутентифікації. Сьогодні приватний замовник в основному готовий платити за безпеку. Завдання розпізнавання осіб актуальна як в області інтелектуальних середовищ, так і в системах безпеки.

Другий важливий аспект – це інтуїтивно зрозумілий інтерфейс. Система була розроблена на мові JavaScript та середовище розробки Visual Code. Була взята архітектура згорткової нейронної мережі MobilNet та були використані деякі функції для покращення роботи системи розпізнавання обличчя.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Face Recognition by Elastic Bunch Graph Matching: CRC Press, ISBN 0-8493-2055-0, Chapter 11, pp. 355-396, (1999) – режим доступу: https://www.face-rec.org/algorithms/EBGM/WisFelKrue99 FaceRecognition-JainBook.pdf.

2. Distortion Invariant Object Recognition in the Dynamic Link Architecture: IEEE TRANSACTIONS ON COMPUTERS, VOL. 42, NO 3, MARCH 1993 – режим доступу: https://www.researchgate.net/publication/3043085_Distortion_Invariant_Object_Recognition_in_the_Dynamic_Link_Architecture.

3. Face Recognition: A Convolutional Neural-Network Approach: IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 8, NO. 1, JANUARY 1997 – режим доступу: https://pdfs.semanticscholar.org/ee1e/322b5f8f15ad3fcd17762fba3da209b0c484.pdf.

4. Face Recognition using Convolutional Neural Network and Simple Logistic Classifier [Електронний ресурс] – режим доступу: http://dap.vsb.cz/wsc17conf/Media/Default/Page/online_wsc17_submission_59.pdf.

5. Face Image Analysis With Convolutional Neural Networks [Електронний ресурс] – режим доступу: https://www.freidok.uni-freiburg.de/fedora/objects/freidok:4835/datastreams/FILE1/content.

6. DeepFace: Closing the Gap to Human-Level Performance in Face Verification – режим доступу: https://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf.

7. Распознавание лиц на групповых фотографиях с использованием алгоритмов сегментации: Франц В.А., Левина О.М., Воронин В.В., Кожин Р.А. Первичная обработка карты глубины изображения. Успехи современной радиоэлектроники. Радиотехника – режим доступу: https://cyberleninka.ru/article/n/raspoznavanie-lits-na-gruppovyh fotografiyah-s-ispolzovaniem-algoritmov-segmentatsii/viewer.

8. Академия Intel. Введение в естественно-интуитивное взаимодействие с

компьютером. Лекция 5: Модуль анализа мимики лица Intel Perceptual Computing SDK [Електронний ресурс] // intuit.ru. 2017. URL: http://www.intuit.ru/studies/courses/10619/1103/lecture/18229

9. Анализ существующих подходов к распознаванию лиц [Електронний ресурс] – режим доступу: https://habr.com/ru/company/synesis/blog/238129/.

10. Face Recognition Vendor Test (FRVT) [Електронний ресурс] – режим доступа: https://www.nist.gov/programs-projects/face-recognition-vendor-test-frvt.

11. Рогозин О.В., Кладов С.А. Сравнительный анализ алгоритмов распознавания лиц в задаче визуальной идентификации МГТУ им. Н.Э. Баумана, Москва, 2013. 8 с.

12. Шерстобитов А.И., Федосов В.П., Приходченко В.А., Тимофеев Д.В. Распознавание лиц на групповых фотографиях с использованием алгоритмов сегментации. Ростов на Дону, 2013. 8 с.

13. Академия Intel. Введение в естественно-интуитивное взаимодействие с компьютером. Лекция 5: Модуль анализа мимики лица Intel Perceptual Computing SDK [Електронний ресурс] // intuit.ru. 2017. URL: http://www.intuit.ru/studies/courses/10619/1103/lecture/18229 .

14. Головко В.А. Нейроинтеллект: Теория и применения. Книга 1. Организация и обучение нейронных сетей с прямыми и обратными связями. – Брест: БПИ, 1999. – 260с.

15. Image-based Face Recognition — Issues and Methods. URL: https://www.face-rec.org/interesting-papers/general/chapter_figure.pdf

16. Face Detection A Survey. URL: https://www.cin.ufpe.br/~rps/Artigos/Face%20Detection%20-%20%20A%20Survey.pdf

17. Об одном подходе к локализации антропометрических точек. URL: https://docplayer.ru/40035730-Ob-odnom-podhode-k-lokalizacii-antropometricheskih-tochek.html

18. The FERET evaluation methodology for face-recognition algorithms. URL: https://ieeexplore.ieee.org/document/879790

19. Design, Implementation and Evaluation of Hardware Vision Systems dedicated to Real-Time Face Recognition. URL: https://www.researchgate.net/publication/221786186_Design_Implementation_and_Evaluation_of_Hardware_Vision_Systems_Dedicated_to_Real-Time_Face_Recognition

20. An Introduction to the Good, the Bad, & the Ugly Face Recognition Challenge Problem. URL: https://ieeexplore.ieee.org/document/5771424

21. DeepFace Closing the Gap to Human-Level Performance in Face Verification.URL: https://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf

22. Taking the bite out of automated naming of characters in TV video. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.180.3756&rep=rep1&type=pdf

23. Towards a Practical Face Recognition System Robust Alignment and Illumination by Sparse Representation. URL: https://www.researchgate.net/publication/224240923_Toward_a_Practical_Face_Recognition_System_Robust_Alignment_and_Illumination_by_Sparse_Representation

24. Алгоритмы обнаружения лица человека для решения прикладных задач анализа и обработки изображений. URL: http://tekhnosfera.com/algoritmy-obnaruzheniya-litsa-cheloveka-dlya-resheniya-prikladnyh-zadach-analiza-i-obrabotki-izobrazheniy

25. Обнаружение и локализация лица на изображении. URL: https://docplayer.ru/25926054-Obnaruzhenie-i-lokalizaciya-lica-na-izobrazhenii.html

26. Разработка и анализ алгоритмов детектирования и классификации объектов на основе методов машинного обучения. URL: https://www.dissercat.com/content/razrabotka-i-analiz-algoritmov-detektirovaniya-i-klassifikatsii-obektov-na-osnove-metodov-ma

27. Overview of the Face Recognition Grand Challenge. URL: https://ieeexplore.ieee.org/document/1467368

28. Об эффективности применения алгоритма SURF в задаче идентификации лиц. URL: https://core.ac.uk/download/pdf/53068349.pdf

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

Файл index.js:

```
const camera = document.getElementById('camera')

const startCamera = () => {
    navigator.mediaDevices.enumerateDevices()
    .then(devices => {
        if (Array.isArray(devices)) {
            devices.forEach(device => {
                if (device.kind === 'videoinput') {
                    if (device.label.includes('')) {
                        navigator.getUserMedia(
                        { video: {
                            deviceId: device.deviceId
                            }},
                            stream => camera.srcObject = stream,
                            error => console.error(error),
                      )  }  } })  } }) }

const loadLabels = () => {

    const labels = ['Andrew', 'Dmitry']

    return Promise.all(labels.map(async label => {
       const descriptions = []
        for (let i = 1; i <= 2; i++) {
            const img = await faceapi.fetchImage(`/assets/lib/face-api/labels/${label}/${i}.jpg`)
            const detections = await faceapi
                .detectSingleFace(img)
                .withFaceLandmarks()
                .withFaceDescriptor()
            descriptions.push(detections.descriptor)
        }
        return new faceapi.LabeledFaceDescriptors(label, descriptions)
    }))

}

Promise.all([
    faceapi.nets.tinyFaceDetector.loadFromUri('/assets/lib/face-api/models'),
    faceapi.nets.faceLandmark68Net.loadFromUri('/assets/lib/face-api/models'),
    faceapi.nets.faceRecognitionNet.loadFromUri('/assets/lib/face-api/models'),
    faceapi.nets.faceExpressionNet.loadFromUri('/assets/lib/face-api/models'),
    faceapi.nets.ageGenderNet.loadFromUri('/assets/lib/face-api/models'),
    faceapi.nets.ssdMobilenetv1.loadFromUri('/assets/lib/face-api/models'),
]).then(startCamera)

camera.addEventListener('play', async () => {
```

```
    const canvas = faceapi.createCanvasFromMedia(camera)
    const canvasSize = {
        width: camera.width,
        height: camera.height
    }
    const labels = await loadLabels()
    faceapi.matchDimensions(canvas, canvasSize)
    document.body.appendChild(canvas)
    setInterval(async () => {
        const detections = await faceapi
            .detectAllFaces(
                camera,
                new faceapi.TinyFaceDetectorOptions()

            )
            .withFaceLandmarks()
            .withFaceExpressions()
            .withAgeAndGender()
            .withFaceDescriptors()
            const resizedDetections = faceapi.resizeResults(detections, canvasSize)

        const faceMatcher = new faceapi.FaceMatcher(labels, 0.6)
        const results = resizedDetections.map(d =>
            faceMatcher.findBestMatch(d.descriptor)
        )
        canvas.getContext('2d').clearRect(0, 0, canvas.width, canvas.height)
        faceapi.draw.drawDetections(canvas, resizedDetections)
        faceapi.draw.drawFaceLandmarks(canvas, resizedDetections)
        faceapi.draw.drawFaceExpressions(canvas, resizedDetections)
        resizedDetections.forEach(detection => {
            const { age, gender, genderProbability } = detection
            new faceapi.draw.DrawTextField([
                `${parseInt(age, 10)} years`,
                `${gender} (${parseInt(genderProbability * 100, 10)})`
            ], detection.detection.box.topRight).draw(canvas)
        })
        results.forEach((result, index) => {
            const box = resizedDetections[index].detection.box
            const { label, distance } = result
            new faceapi.draw.DrawTextField([
                `${label} (${parseInt(distance * 100, 10)})`
            ], box.bottomRight).draw(canvas)
        })
    }, 100)
})
```

Файл index.php:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="icon" href="./assets/img/favicon.ico"/>
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Face Detection by DK</title>
    <link rel="stylesheet" href="./assets/css/index.css">
    <link rel="stylesheet" href="./assets/css/style.css">
</head>
<body>
<div class="infoblock">
<div class="backlogo">
</div>
<div class="textmain">
Developed by<br>Dmitry Kornienko
</div>
</div>
    <video
    autoplay
    id="camera"
    width="720"
    height="560"
    ></video>
    <script src="assets/lib/face-api/face-api.min.js"></script>
    <script src="assets/js/index.js"></script>

<script src="assets/js/app.js"></script>

<script src="assets/lib/stats.js"></script>
</body>
</html>
```

Файл style.css:

```
html,body{
  z-index: -1;
                                        width:100%;
                                        height:100%;
  background-image: url(/assets/img/bg.png);
  position: absolute;
}
body{
  font:normal 75% Arial, Helvetica, sans-serif;
}
canvas{
  display:block;
}
.infoblock{
                                        width:400px;
                                        height:120px;
                                        color:#fff;
                                        position:absolute;
                                        top:40px;
                                        border:5px solid #fff;
                                        border-radius:15px;
                                        text-align:center;
                                        font:26px impact;
  background-image: url(/assets/img/bg2.png);
```

```
}
.backlogo{
                                        background-image: url(/assets/img/favicon3.png);
                                        background-size:100px;
                                        background-repeat:no-repeat;
                                        float:left;
                                        width:150px;
                                        height:150px;
                                        margin:10px 0 10px 10px;
}
.textmain{
                                        padding-top:30px;
                                        padding-right:50px;
}
```

ДОДАТОК Б

ВІДГУК

керівника економічного розділу

на кваліфікаційну роботу магістра

на тему: «Методи, алгоритми та інформаційна технологія розпізнавання обличчя на основі згорткової нейронної мережі»

студента групи 122м-19-1 Корнієнка Дмитра Ігоровича

Керівник економічного розділу

доцент каф. ПЕП та ПУ, к.е.н. **Л. В. Касьяненко**

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файла	**Опис**
Пояснювальні документи	
Диплом_Корнієнко.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_Корнієнко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
AIFaceDetection.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_Корнієнко.ppt	Презентація до кваліфікаційної роботи