МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА СИСТЕМИ НЕЙРОМЕРЕЖЕВОГО РОЗПІЗНАВАННЯ ОБЛИЧЧЯ»**

Виконав: студент 2 курсу, групи 8.1212-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
Т. Беккаві
(ініціали та прізвище)

Керівник старший викладач кафедри програмної інженерії, PhD Чопорова О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Беккаві Таріку
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка системи нейромережевого розпізнавання обличчя

керівник роботи Чопорова Оксана Володимирівна, PhD
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 642-с

2. Строк подання студентом роботи 26.02.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Аналіз задачі ідентифікації обличчя людини.
4. Проєктування автоматизованої системи ідентифікації обличчя людини.
5. Реалізація та тестування автоматизованої системи ідентифікації обличчя людини.

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 03.05.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	19.05.2023	
2.	Збір вихідних даних.	23.06.2023	
3.	Обробка методичних та теоретичних	04.09.2023	
	джерел.		
4.	Розробка першого та другого розділу.	04.12.2023	
5.	Розробка третього розділу.	22.01.2024	
6.	Оформлення та нормоконтроль	19.02.2024	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	13.03.2024	

Студент _____ Т. Беккаві
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Чопорова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка системи нейромережевого розпізнавання обличчя»: 95 с., 42 рис., 14 табл., 22 джерела, 1 додаток.

АВТОМАТИЗАЦІЯ, БІБЛІОТЕКА, НАВЧАННЯ, НЕЙРОМЕРЕЖА, ОКО, РОЗПІЗНАВАННЯ.

Об’єкт дослідження – роботу комп'ютерного зору, як працює нейромережа для розпізнавання обличчя.

Мета роботи: визначення можливостей при реалізації системи розпізнавання облич людини за допомогою бібліотек комп’ютерного зору та розпізнавання облич.

Метод дослідження – даної роботи є технології ідентифікації особистості на основі зображення особи, методи детектування, розпізнавання, формування орієнтирів та кодування ознак облич.

В дипломній роботі була розроблена програма для розпізнавання облич, в автоматичному режимі. По закінченню розробки та тестування, вірогідність розпізнання обличчя варіюється між 70%–90%.

SUMMARY

Master’s qualifying paper «Development of a Neural Network Faces Recognition System»: 95 pages, 42 figures, 14 tables, 22 references, 1 supplement.

AUTOMATION, LIBRARY, LEARNING, NEURAL NETWORK, EYE, RECOGNITION.

The object of research is the work of computer vision, how a neural network works for face recognition.

The purpose of the work: to determine the possibilities in the implementation of the human face recognition system using computer vision and face recognition libraries.

The method of research – this work includes the technologies of personal identification based on the image of a person, methods of detection, recognition, formation of landmarks and coding of facial features.

In the thesis, a program was developed for face recognition, in automatic mode. After development and testing, the probability of facial recognition varies between 70%–90%.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 7
1 Аналіз задачі ідентифікації обличчя людини 8
1.1 Поняття та застосування задачі ідентифікації обличчя людини 8
1.2 Аналіз наявних програмних засобів 14
1.3 Аналіз наявних технологій 22
1.3.1 Розпізнавання облич на основі метода Віоли-Джонса 28
1.3.2 Штучні нейронні мережі 32
1.3.3 Опис нейронної мережі глибокого навчання 43
1.4 Висновки до розділу 1 51
2 Проєктування автоматизованої системи ідентифікації обличчя людини 52
2.1 Розробка архітектури та структури програмного комплексу 52
2.2 Опис архітектури нейронної мережі 58
2.3 Проєктування бази даних та інтерфейсу системи 60
2.4 Висновки до розділу 2 63
3 Реалізація та тестування автоматизованої системи ідентифікації обличчя людини 64
3.1 Реалізація функціоналу системи 64
3.2 Процес побудови та тренування моделі 68
3.3 Тестування програмного комплексу 71
3.4 Висновки до розділу 3 78
Висновки 79
Перелік посилань 80
Додаток А Основний код 83

ВСТУП

У даний час методи аналізу даних активно розвиваються в напрямку обробки великих обсягів даних. Джерелами виникнення великих даних є дані, що безперервно надходять з вимірювальних пристроїв (аудіо і відео-реєстрації та т.п.).

Обробка зображень сьогодні широко використовується в системах безпеки для ідентифікації людей за зображеннями осіб, моніторингу стану технічних об'єктів.

Технологія ідентифікації особистості на основі зображення особи, зважаючи на інші біометричні показники, не вимагає фізичного контакту з пристроєм та враховуючи стрімкий розвиток цифрової техніки є найбільш прийнятною для масового застосування.

Метою даної роботи є визначення можливостей при реалізації системи розпізнавання облич людини за допомогою бібліотек комп’ютерного зору та розпізнавання облич.

Об’єктами досліджень даної роботи є технології ідентифікації особистості на основі зображення особи, методи детектування, розпізнавання, формування орієнтирів та кодування ознак облич.

У результаті виконання роботи буде створена система, здатна розпізнати обличчя людини.

1 АНАЛІЗ ЗАДАЧІ ІДЕНТИФІКАЦІЇ ОБЛИЧЧЯ ЛЮДИНИ

1.1 Поняття та застосування задачі ідентифікації обличчя людини

Розпізнавання облич є галуззю, яка швидко розвивається протягом останнього десятиліття. Її прогрес в значній мірі пояснюється областями застосування; Розваги, інформаційна безпека, відеоспостереження і використання правоохоронними органами – це всього лише декілька областей, де технологія розпізнавання облич може застосовуватися з великою ефективністю [11]. Використання відбитків пальців та сканування райдужної оболонки ока – це два дуже точних методи біометричної ідентифікації, але обидва вимагають прямої взаємодії та контакту з користувачем [12]. Розпізнавання облич з іншого боку, є методом біометричної ідентифікації, який може бути використаний без прямої взаємодії з користувачем, коли суб'єкти не знають, що вони в даний час ідентифікуються.

Заміна комп'ютерних паролів, PIN-кодів або паспортного контролю на зображення або відео облич може мати найсерйозніші зміни в повсякденному житті. Розпізнавання облич, по суті, використовується для проходження паспортного контролю на кордоні Китаю і Гонконгу з 2005 року, де більш ніж 400000 людей перетинають кордон кожен день [13]. Це ознака того, що ця технологія стає ближче до того, щоб застосовуватися повсюдно, але все ще не повністю готова. Нові відкриття у суміжних дисциплінах, таких як обробка зображень, розпізнавання образів, машинне навчання та комп'ютерний зір, комп'ютерна графіка та психологія доволі часто застосовуються та об'єднуються у сфері розпізнавання облич [11].

Хоча перша система автоматичного розпізнавання облич була запропонована в 1973 році, вона не була реалізована до початку 1990-х років,

коли дослідження у цій сфері стали популярними. Плідна робота про алгоритм Eigenfaces там представлення облич у низьковимірних просторах зіграла важливу роль у популяризації даної теми [13], так як на той час це був новий підхід до проблеми розпізнавання облич який багато людей визнало цікавим [14]. Інші фактори, такі як збільшення обчислювальних потужностей та розробка різноманітних доступних баз облич також допомогли розвитку цієї теми.

Поява більшої кількості анотованих даних, на яких можна було тестувати та тренувати алгоритми, а також розробка фіксованих протоколів оцінки ефективності алгоритмів зробили процес порівняння методів та результатів набагато простішим. Більшість досліджень у сфері розпізнавання облич історично були зроблені на чорно-білих базах зображень з контрольованим відхиленням. Типовими прикладами таких баз є FERET [15] (1199 людей були сфотографовані у різних контрольованих позах та умовах освітлення), Єльська база [16] (15 суб'єктів в 11 різних конфігураціях) і база даних BioID [17] (1521 фотографія 23 суб'єктів без обмежень у позі). Пізніше дослідження стали зосереджуватися на розпізнаванні облич у середовищах без обмежень у виразі обличчя, одязі, освітленні, положенні тіла та фоні. Набір даних “Labeled faces in the wild” (LFW) [18] був створений використовуючи зображення знайдені в Інтернеті та дуже широко використовувався для вимірювань продуктивності у середовищах без обмежень так як зображення у ньому не є синтетичними та є дуже різними. Дослідження розпізнавання облич з використанням інших типів вхідних даних, таких як зображення з високою роздільною здатністю і 3D-сканування [19], спектральні зображення [12] і відео [20] також стали більш поширеними. Усі типи вхідних даних мають свої плюси, наприклад більшу якість інформації, але також вони мають відповідні проблеми.

Задачу розпізнавання облич можна розглядати як нелінійну проблему розпізнавання образів [12]. Вона може бути поділена на задачу підтвердження обличчя, коли зображення обличчя порівнюється з іншим зображенням обличчя,

і визначає чи на них зображена одна й та сама людина, та на задачу ідентифікації обличчя, коли співпадіння шукається серед набору з N зображень, що були збережені до того [13]. При відсутності попереднього знання про те, що може бути розглянуто як релевантна інформація під час розпізнавання облич, це є дуже складною проблемою, особливо зважаючи на те що всі зображення були взяті при різних умовах. Янг та інші [21] перераховують різний масштаб, позицію голови, вираз обличчя, умови освітлення, характеристики камери, особливості зовнішності (борода, вуса, окуляри) та часткове перекриття обличчя як складності у загальному випадку виділення обличчя. На жаль, цей самий список проблем є й в розпізнавання облич. Абате та інші [12] також розглядають час як фактор, що може вплинути на продуктивність, так як старіння може сильно змінити зовнішність людини.

Складність проблеми зростає, якщо розглядати повністю автоматизовані системи. Проблема розпізнавання облич буде розширена на декілька підзадач, які можуть залежати від метода вибору. Спільним у всіх цих методів є те, що всі вони потребують одне або декілька зображень обличчя, тож відповідно першим кроком є виділення будь-яких облич на зображенні. Обличчя потім вирізається з зображення та переробляється у більш зручному поданні, що зазвичай включає нормалізацію щодо освітлення або геометричних перетворень, наприклад зсувів або поворотів. Це робиться для збільшення продуктивності системи у неконтрольованих умовах та зменшення можливих варіацій освітлення та поз, що можуть бути [13]. Після нормалізації останній крок до розпізнавання облич може бути додатково поділений на подання та класифікацію ознак [22]. Ознаки виділяються з обличчя для того щоб полегшити завдання знаходження збігів у базі облич. Зазвичай вони створюються за допомогою кодування інформації про обличчя у більш компактну та виразну форму використовуючи дескриптор, що в оптимальному випадку має бути незмінним для всіх варіантів зображень. Приклади виділення ознак включають локальні бінарні шаблони, гістограми

орієнтованих градієнтів, вейвлети Габора та Scale-invariant feature transform (SIFT) дескриптори [23, 34]. Далі до ознаки застосовується класифікатор для того, щоб знайти найбільший збіг серед усіх інших збережених облич, які також були оброблені таким чином. Дослідники постійно вдосконалюють продуктивність їхніх систем, додаючи нові підходи та змінюючи методи.

В сучасних умовах проблема автентифікації та ідентифікації біометричних даних є актуальною в обчислювальних мережах, електронних системах управління, електронній комерції і взагалі там, де є необхідність переконатись у справжності отриманого каналами зв'язку або на машинних носіях повідомлення чи документа.

Загальна схема розпізнавання складається із 6 пунктів (рис. 1.1).

Рисунок 1.1 – Схема процесу розпізнавання

Виділяють два етапи розпізнавання обличчя. Перший етап полягає у виділенні ознак, а другий – класифікує об’єкти.

На кожному обличчі є безліч відмінних орієнтирів, різні вершини і западини, що становлять риси обличчя. При розпізнаванні визначаються ці орієнтири як вузлові точки. Кожна людська особа має приблизно 80 вузлових точок. Ось деякі з них, використовувані у спеціальному програмному забезпеченні:

– відстань між очима;
– ширина носа;
– глибина очниць;
– форма вилиць;
– довжина лінії щелепи.

Ці вузлові точки вимірюються, створюючи числовий код, званий «відбитком обличчя», який представляє обличчя в базі даних [1].

У минулому програмне забезпечення для розпізнавання осіб покладалось на 2D-зображення для порівняння або ідентифікації іншого 2D-зображення з бази даних. Щоб бути ефективним і точним, на знімку повинно було бути обличчя, яке дивилося майже прямо в камеру, з невеликим відхиленням світла або виразу обличчя від зображення в базі даних. Це створило справжню проблему [1].

У більшості випадків зображення не були зроблені в контрольованому середовищі. Навіть найменш значні зміни в освітленні або орієнтації можуть знизити ефективність системи, через що вони не зможуть бути порівняні ні з однією особою бази даних, що призводить до високого рівня помилок.

Серед методів біометричного розпізнавання, що вже є звичайними, найперспективнішим є розпізнавання людини за обличчям. Дана методика при собі має перелік переваг поміж більшості аналогічних: з високою точністю визначення алгоритм дає можливість перевіряти на певній відстані, допускає анонімний аналіз та потребує тільки наявність відеокамери. Чисельна кількість алгоритмів, що окрім швидкості та точності знаходження дають системі можливість виконувати роботу при різних обставинах. Набір таких

характеристик спонукав розвивати метод, доки він не став найпоширенішим після дактилоскопії.

Об’єднання декількох алгоритмів для аналізу обличчя необхідне рішення для покращення точності. Для прикладу, ідентифікація обличчя доповниться ідентифікацією за вушною раковиною, що і так гарно себе показує на практиці. Не потрібно забувати що погана оптимізація при використанні великого об’єму алгоритмів, нівелює переваги від використання комбінацій алгоритмів

Найперспективніша тенденція ринку біометрії – це вихід на ринок інтелектуальних цифрових камер, з реалізованою функцією аналізування облич за допомогою інтегрованої логіки. Такі камери, окрім якісного зображення, мають можливість пов’язати із зображенням метадані з відомостями про виявлені обличчя. Це розвантажує апаратні потужності, що, натомість, зменшує ціну систем біометричного розпізнавання та робить їх більш доступними. Також, розвантажуються канали передачі даних, оскільки передаються стиснені дані і незначний потік виявлених зображень облич.

Системи розпізнавання облич сьогодні використовуються не тільки для робіт на кшталт виявлення злочинців чи осіб у розшуку в місцях скупчення людей, а й для звичайних домашніх справ. Через доступність камер, удосконалення та оптимізацію алгоритмів виявлення і аналізу облич, підвищилась точність розпізнавання. Більш звичним на ринку біометричного аналізу стає контроль доступу до персональних пристроїв за обличчям користувача. Статистика програм показує їх надійність та зручність роботи з ними. Характеристики таких програм не відрізняються суттєво, тому вибір залежить від бажання користувача.

Якісне розпізнавання обличчя може відбутися тільки за певних умов. Тому, перед впровадженням біометрії для обличчя, треба визначити, якими будуть умови експлуатації такої системи. Більшість сучасних систем розпізнавання реально забезпечити такими умовами, щоб системи добре функціонували.

Наприклад на пропускних пунктах слід організувати потік людей, для можливості короткочасної фіксації облич осіб. Фронтальне положення камер відносно облич не повинно перевищувати 30 град.

Дотримання таких умов – вірний ключ для правильної ідентифікації та пошуку людей, до яких є інтерес, при чому, з максимальною точністю, що заявлена виробниками таких систем.

1.2 Аналіз наявних програмних засобів

3D-розпізнавання. Нещодавно з’явилася тенденція в програмному забезпеченні для розпізнавання облич, що використовує 3D-модель, яка, як стверджується, забезпечує більшу точність. Для отримання тривимірного зображення обличчя людини в режимі реального часу, методика 3D-розпізнавання використовує відмінні риси обличчя – місця, де тверді тканини і кістки найбільш помітні, такі як вигини очниці, носа і підборіддя, – щоб ідентифікувати предмет. Всі ці області унікальні і не змінюються з часом (рис. 1.2).

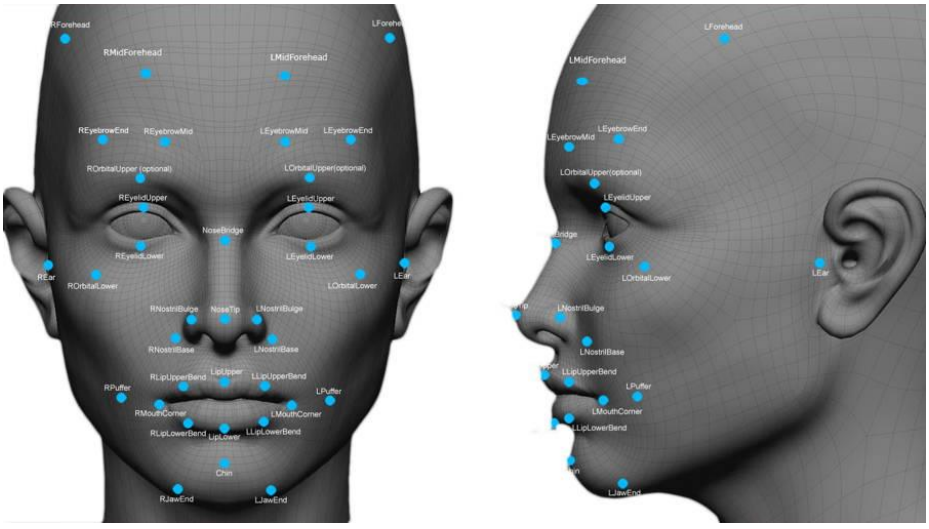


Рисунок 1.2 – Тривимірна модель обличчя

Використовуючи глибину і вісь вимірювання, на які не впливає освітлення, 3D-розпізнавання облич можна використовувати навіть у темряві і є можливість розпізнавати об'єкт під різними кутами огляду з можливістю розпізнавання до 90 градусів (обличчя в профіль) [1].

Використовуючи програмне забезпечення з можливостями 3D розпізнавання, система виконує ряд кроків для перевірки особистості людини [1]. Розглянемо їх.

Виявлення – зображення може бути отримане шляхом цифрового сканування існуючої фотографії (2D) або за допомогою відеозображення для отримання живого зображення об'єкта (3D) [1].

Вирівнювання – після виявлення особи система визначає положення, розмір і позу голови. Як зазначалося раніше, об'єкт може бути розпізнаний під кутом до 90 градусів, у той час як в режимі 2D голова повинна бути повернута до камери як мінімум на 35 градусів.

Вимірювання – система вимірює вигини особи в субміліметровому (або мікрохвильовому) масштабі і створює шаблон.

Подання – система переводить шаблон в унікальний код. Це кодування дає кожному шаблоном набір чисел для позначення рис обличчя об’єкта.

Відповідність – якщо зображення є тривимірним, а база даних містить тривимірні зображення, то порівняння буде відбуватися без внесення будь-яких змін у зображення. Однак в даний час існує проблема з базами даних, які все ще представлені у вигляді 2D-зображень. 3D – це живий, мінливий об’єкт у порівнянні з плоским стабільним зображенням. Нові технології вирішують цю проблему. Коли робиться тривимірне зображення, ідентифікуються різні точки (зазвичай три). Наприклад, зовнішня частина ока, внутрішня частина ока і кінчик носа будуть витягнуті і виміряні. Як тільки ці вимірювання будуть виконані, до зображення буде застосований алгоритм (покрокова процедура) для перетворення його в 2D-зображення. Після перетворення програма порівняє

зображення з 2D-зображеннями в базі даних, щоб знайти можливий збіг.

Перевірка або ідентифікація – при перевірці зображення зіставляється тільки з одним зображенням в базі даних (1:1). Наприклад, отримане зображення об'єкта можна зіставити із зображенням в базі даних Міністерства Внутрішніх справ, щоб переконатися, що об'єкт є тим, ким він себе називає. Якщо метою є ідентифікація, то зображення порівнюється з усіма зображеннями в базі даних, що дає оцінку для кожного потенційного збігу (1:N). У цьому випадку можна взяти зображення і порівняти його з базою даних знімків, щоб визначити, хто є об'єктом.

Біомедичне розпізнавання. Зображення не завжди можна перевірити чи ідентифікувати тільки за допомогою розпізнавання облич. Компанія Identix® створила новий продукт для підвищення точності. При розробці FaceIt®Argus використовуються біометричні дані шкіри, унікальна текстура шкіри, що дозволяє отримувати ще більш точні результати [1].

Процес, званий аналізом текстури поверхні, працює так само, як і розпізнавання обличчя. Робиться знімок ділянки шкіри, званого відбитком шкіри. Потім ця ділянка розбивається на більш дрібні блоки. Використовуючи алгоритми, здатні перетворювати ділянку шкіри в математично вимір простір, система потім розпізнає будь-які лінії, пори і фактичну текстуру шкіри. Він може ідентифікувати відмінності між ідентичними близнятами, що поки неможливо з використанням тільки програмного забезпечення для розпізнавання облич. Згідно Identix, за рахунок об'єднання розпізнавання облич з аналізом текстури поверхні точність ідентифікації може підвищитися на 20 – 25 відсотків [1].

У даний час FaceIt використовує три різні шаблони для підтвердження або ідентифікації об'єкта: вектор, аналіз локальних особливостей і аналіз текстури поверхні.

Векторний шаблон дуже малий і використовується для швидкого пошуку по всій базі даних, у першу чергу для пошуку «один до багатьох».

Шаблон аналізу локальних об'єктів (LFA) виконує вторинний пошук упорядкованих збігів, наступних за векторним шаблоном.

Аналіз текстури поверхні (STA) – найбільший з трьох. Він виконує останній прохід після пошуку за шаблоном LFA, покладаючись на особливості шкіри на зображенні, яке містить найбільш повну інформацію.

Завдяки поєднанню всіх трьох шаблонів FaceIt® має перевагу перед іншими системами. Він нечутливий до змін у виразі обличчя, включаючи моргання, похмурий погляд або посмішку, і здатний компенсувати наявність вусів або бороди і зовнішній вигляд окулярів. Система також незалежна від ознак раси і статі [1].

Однак це не ідеальна система. Є кілька факторів, які можуть перешкодити розпізнаванню, у тому числі:

– значні відблиски на окулярах або наявність сонячних окулярів;
– довге волосся, що закриває центральну частину обличчя;
– погане освітлення, через яке обличчя буде погано освітлене;
– низька роздільна здатність (зображення було знято занадто далеко).

Identix – не єдина компанія, яка пропонує системи розпізнавання осіб. Хоча більшість з них працює так само, як FaceIt, є кілька варіацій. Наприклад, компанія під назвою Animetrix, Inc. має продукт під назвою FACEngine ID® SetLight, який може коригувати умови освітлення, що зазвичай не дозволяють якісно виявити обличчя, знижуючи ризик помилкових збігів. У Sensible Vision, Inc. є продукт, який може захистити комп'ютер за допомогою розпізнавання обличчя. Комп'ютер буде вмикатися і залишатися доступним тільки до тих пір, поки потрібний користувач знаходиться перед екраном. Як тільки користувач йде із зони прямої видимості, комп'ютер автоматично блокується від інших користувачів.

Завдяки цим технологічним досягненням системи розпізнавання облич і шкіри отримали більш широке поширення, ніж всього кілька років тому.

У минулому основними користувачами програмного забезпечення для розпізнавання осіб були правоохоронні органи, які використовували такі системи для захоплення випадкових осіб у натовпі. Деякі урядові агентства також використовували ці системи для забезпечення безпеки та запобігання шахрайства з виборцями. Уряд США розпочав програму під назвою US-VISIT (Технологія визначення статусу відвідувача і іммігранта в США), призначену для іноземних мандрівників, які отримують дозвіл на в'їзд у Сполучені Штати. Коли іноземний мандрівник отримує візу, він здає відбитки пальців і фотографується. Відбитки пальців і фотографії перевіряються по базі даних відомих злочинців і підозрюваних у тероризмі. Коли мандрівник прибуває в Сполучені Штати в пункт пропуску, ті ж відбитки пальців і фотографії будуть використовуватися для підтвердження того, що особа, яка отримала візу, є тією самою особою, яка намагається отримати в’їзд.

Однак зараз набагато більше ситуацій, коли програмне забезпечення стає популярним. У міру того, як системи стають дешевшими, їх використання стає більш поширеним. Тепер вони сумісні з камерами і комп'ютерами, які вже використовуються в банках і аеропортах. TSA в даний час працює і тестує свою програму для зареєстрованих мандрівників. Програма забезпечить швидку перевірку безпеки пасажирів, що добровільно надають інформацію, і виконає оцінку загроз безпеки. У аеропорту зареєстрованим мандрівникам будуть відведені спеціальні черги, які будуть рухатися швидше, перевіряючи мандрівника за рисами обличчя.

Інші потенційні додатки включають у себе безпеку банкоматів і переведення в готівку чеків. Програма здатна швидко перевірити особу клієнта. Після згоди клієнта банкомат або кіоск з переведення в готівку чеків зберігає його цифрове зображення. Потім програмне забезпечення FaceIt створює відбиток особи фотографії, щоб захистити клієнтів від крадіжки особистих даних і шахрайських транзакцій. При використанні програмного забезпечення для

розпізнавання облич відпадає необхідність у посвідченні особи з фотографією, банківській картці або особистому ідентифікаційному номері (ПІН) для підтвердження особи клієнта. Таким чином підприємства можуть запобігти шахрайству.

Хоча всі наведені вище приклади працюють з дозволу людини, не всі системи використовуються за згодою. Іноді системи фотографують всіх відвідувачів без їхнього відома і дозволу. Противники систем відзначають, що, хоча в деяких випадках вони дійсно забезпечують безпеку, цього недостатньо, щоб виправдати відчуття пригнічення свободи. Багато хто вважає, що використання цих систем є занадто серйозним порушенням конфіденційності, але на цьому їхні побоювання не закінчуються. Вони також вказують на ризик, пов'язаний з крадіжкою особистих даних. Навіть корпорації по розпізнаванню облич визнають, що чим ширше використовується технологія, тим вище ймовірність крадіжки особистих даних або шахрайства.

Як і в багатьох технологіях що розвиваються, неймовірний потенціал розпізнавання облич має деякі недоліки, але виробники прагнуть покращити зручність використання і точність систем.

Також системи розпізнавання застосовують у розумних камерах для відеоспостереження. Загальною ідеєю існуючих рішень є обробка відеопотоку шляхом застосування набору алгоритмів, які можуть вирішувати наступні задачі [27]:

– збільшення відношення сигнал/шум;
– визначення відстані до об’єктів;
– розпізнавання об’єктів;
– визначення здійснюваних особами дій [27].

Пропоновані ринком продукти можна розділити у дві категорії, комплексні системи відеоспостереження і розумні камери [27].

Рішення, описані в цьому розділі, присвячені комерційним системам, які передбачають автоматичне виявлення визначених подій на основі обробки відео. Можна придбати або програмне забезпечення, яке постачається, або обладнанням, що таке забезпечення використовує. Загалом такі системи включають кілька основних пристроїв, для збору, перетворення та зберігання даних. Додатково програмне забезпечення кінцевого користувача постачається або як вебінтерфейс, або, як правило, як спеціалізована програма, що спрощує обслуговування та управління системою. Обробка відео реалізована на основі передових технологій, розроблених компаніями. У деяких випадках пропонуються пристрої з чорним ящиком, до якого підключається кожна камера (одна або більше). Як альтернатива, виділяються спеціалізовані високопродуктивні сервери для підтримки аналізу декількох відеопотоків [27].

Програмні продукти включають загальні характеристики системи, а також різні конфігурації. Типовими недоліками систем цього типу є вартість, необхідність встановлення великої кількості додаткового обладнання, неможливість додавання компонентів від сторонніх розробників, закритий програмний код [27].

PC NVR – це гнучке та економічно вигідне рішення, яке може керувати камерами IP / Megapixel. NVR використовує технологію відкритої платформи, яка підтримує до 52 брендів IP / Megapixel камер з більш ніж 1100 моделями. Інші зручні інструменти включають інтуїтивно зрозумілий графік запису, засоби покращення відео та мобільну підтримку [27].

NVR підтримується автентичною центральною системою управління, яка є справжнім рішенням для моніторингу та управління, яке підтримує необмежену кількість камер. Основна консоль – це сервер запису NVR. Він може відображати відео в прямому ефірі та налаштовувати систему. NVR має інтуїтивний дизайн, який можна вивчити дуже швидко. Основні функції включають наступні [27]:

- потокове передавання декількох відео в реальному часі з IP-камер та IP-відео серверів [27];
- унікальний графік запису графічного інтерфейсу [27];
- 10 варіантів реагування та повідомлення під час тривоги (наприклад, повідомлення електронною поштою та спливаюче вікно);
- повторне відображення каналу (основна консоль може відображати відео в реальному часі та налаштовувати диктофон, система відтворення робить інтелектуальний і швидший перегляд записів за допомогою інтелектуального пошуку) [27].

Збереження відео перетворює зображення у поширені формати відео. Інструмент покращення відео можна використовувати для різкості, яскравості або навіть сірого масштабу зображення. Різні журнали ведуть облік усіх подій. Проста у використанні багатоканальна програма резервного копіювання для архівації аудіо та відео локально або віддалено. Система резервного копіювання також може робити знімки записів і зберігати їх в окремому місці. Іншим підходом до застосування відеоаналітики в системах спостереження є використання так званих смарт-камер. Розумна камера – це звичайна камера, оснащена додатковим модулем, де виконується обробка відео. Ці два елементи, як правило, інтегровані всередині одного корпусу. Такі камери можна використовувати у вже існуючих системах спостереження, що пропонують оператору людини додаткову інформацію, отриману з модуля відеоаналітики. Часто поряд з цим рішенням виробники постачають спеціальне програмне забезпечення для камер з інтерфейсом, що дозволяє спілкуватися з користувачем. Таким чином оператор може бути адекватно оповіщений, відображаючи додаткову інформацію про поточний відеопотік. Типовими недоліками систем даного типу є обмежений функціонал, неможливість розширення програмних і апаратних засобів, обробка даних у корпоративних «хмарах» [27].

Samsung Techwin, провідна у світі технологія візуалізації, відіграє важливу роль у захисті безпеки та щастя людей, забезпечуючи комплексний асортимент продуктів та комплексних рішень, починаючи від спостереження за містом, захищаючи вулиці, аеропорти, порти, промислові об'єкти, військові установки тощо.

Samsung Techwin sets – новий орієнтир на внутрішньому та міжнародному ринку безпеки, забезпечуючи більш високу якість, більш чисті зображення та найсучасніші мережеві функції. Samsung Techwin має на меті забезпечити безпечне рішення, яке задовольнятиме потреби користувачів і зараз, і в майбутньому, і прагне стати світовим провідним постачальником професійних рішень з безпеки [27]. Переваги:

- відео у розширенні 1080p;
- миттєві сповіщення про активність;
- двостороннє аудіо;
- робота у темряві;
- можливість налаштування зон активності [27].

1.3 Аналіз наявних технологій

Вибір категорії та методу для розпізнавання залежить від обмежень і умов завдання розпізнавання осіб. У якості обмежень, що впливають на вибір методу розв'язання задачі, слід виділити:

- наявність або відсутність обмежень на можливі штучні перешкоди на обличчі;
- просторові характеристики становища осіб;
- кольоровість зображення;
- масштаб осіб і роздільна здатність зображення;

– кількість осіб на зображенні;
– умови освітленості об'єктів;
– пріоритет в мінімізації помилкових розпізнавань або в кількості розпізнаних осіб.

Існує велика кількість методів і підходів, що використовуються в системах розпізнавання осіб [2]. Серед них можна виділити метод головних компонент (МГК), лінійний дискримінантний аналіз (ЛДА), сховані марковські моделі (СММ), вейвлети Габора. При використанні прихованих марковських моделей [3] для вирішення задачі розпізнавання осіб для кожного класу осіб обчислюється своя прихована марковська модель. Далі для невідомого способу запускаються всі наявні моделі, і серед них шукається та, яка видає найближчий результат. недоліком такого підходу є те, що приховані марковські моделі не мають гарної роздільної здатності, тому що алгоритм навчання максимізує відгук на свої класи, але не мінімізує відгук на інші класи.

Методи розпізнавання, засновані на використанні вейвлетів Габора [4, 5], показують високу ефективність. Фільтри Габора використовуються на стадії попередньої обробки для формування вектора Габор-особливостей зображення особи. Метод вейвлетів Габора стійкий до змін в освітленні, оскільки не використовує безпосередньо значення відтінків сірого кожного пікселя, а витягує особливості.

Далі в роботі описуються і аналізуються сучасні методи розпізнавання осіб.

Метод головних компонентів (Principal Component Analysis, PCA). Ідея методу полягає в поданні зображень облич у вигляді набору (Вектора) головних компонентів зображень, званих «власні особи. Вони, особи, мають корисну властивість: що зображення, відповідне кожному такому вектору, має подібну до обличчя форму [4].

Обчислення головних компонентів зводиться до обчислення власних векторів і власних значень коваріаційної матриці, яка розраховується з

зображення. Сума головних компонент, помножених на відповідні власні вектора, є реконструкцією зображення [2].

Для кожного зображення особи обчислюються його головні компоненти. Зазвичай береться від 5 до 200 головних компонентів. процес розпізнавання полягає в порівнянні головних компонент невідомого зображення з компонентами всіх відомих зображень. При цьому передбачається, що зображення осіб, відповідних одній людині, згруповані в кластери у власному просторі. З бази даних вибираються зображення-кандидати, які мають найменшу відстань від вхідного (невідомого) зображення [2].

Метод власних осіб вимагає для свого застосування ідеалізованих умов таких, як єдині параметри освітленості, нейтральний вираз обличчя, відсутність перешкод на зразок окулярів і борід. При недотриманні цих умов головні компоненти не будуть відображати міжкласові варіації. Наприклад, при різних умовах освітленості метод власних осіб практично непридатний, оскільки перші головні компоненти переважно відображають зміни освітлення, і порівняння видає зображення, що мають схожий рівень освітленості. При дотриманні ідеалізованих умов точність розпізнавання з використанням даного методу може досягати значення понад 90%, що є дуже хорошим результатом. Обчислення набору власних векторів відрізняється високою трудомісткістю. Один із способів – це згортка зображень по рядках і стовбцях – у такій формі представлення зображення має на порядок менший розмір, обчислення та розпізнавання відбувається швидше, але відновити вихідне зображення вже неможливо [2].

Метод Віоли-Джонса. Даний метод є високоефективним для пошуку об'єктів на зображеннях і відеопослідовностях в режимі реального часу [3, 4]. Цей детектор має вкрай низьку ймовірністю помилкового виявлення особи. Метод добре працює і виявляє риси обличчя навіть при спостереженні об'єкта під невеликим кутом, приблизно до 30 °. Точність розпізнавання з використанням даного методу може досягати значення понад 90%, що є дуже хорошим

результатом. при вуглі нахилу більше 30 ° ймовірність виявлення особи різко падає. Зазначена особливість методу не дозволяє в стандартній реалізації детектувати обличчя людини, повернене під довільним кутом, що значною мірою ускладнює або робить неможливим використання алгоритму в сучасних виробничих системах з урахуванням їх зростаючих потреб.

Порівняння шаблонів (Template Matching). Основа цього методу полягає у виділенні областей особи на зображенні, і подальшому порівнянні цих областей для двох різних зображень. Кожна схожа область збільшує міру схожості зображень. Для порівняння областей використовуються найпростіші алгоритми на кшталт попіксельного порівняння [2].

Недолік цього методу полягає в тому, що він вимагає багато ресурсів як для зберігання ділянок, так і для їх порівняння. З огляду на те, що використовується найпростіший алгоритм порівняння, зображення повинні бути зняті в суворо встановлених умовах: не допускається помітних змін ракурсу, освітлення, емоційного вираження і ін. Точність розпізнавання з використанням даного методу складає близько 80%, що є гарним результатом [2].

Нейронна мережа Хопфілда. Алгоритм навчання мережі Хопфілда істотно відрізняється від класичних алгоритмів навчання перептронів тим, що замість послідовного наближення до потрібного стану з обчисленням помилок, всі коефіцієнти вагової матриці розраховуються за однією формулою, за один цикл, після чого мережа відразу готова до роботи.

Обмеження методу:

– запам’ятовуванні образи не повинні бути сильно схожі;
– зображення не повинно бути зміщене або повернуте щодо його вихідного стану.

Для усунення цих недоліків розглядаються різні модифікації класичної нейронної мережі Хопфілда. Це мережа з ортогональним перетворенням, що дозволяє відновлювати сильно скорельовані образи шляхом перетворення їх

вихідної множини до дуальної множини векторів. Таким чином, виходить нейронна мережа, яка може запам'ятовувати кілька векторів, і при подачі на вхід будь-якого вектора, може визначити, на якій з запам’ятованих він найбільше схожий.

Точність розпізнавання з використанням даного методу складає понад 90%, а в ряді випадків – навіть наближається до 100%, що є майже відмінним результатом.

Серед сучасних робіт, присвячених розпізнаванню осіб, можна відзначити статтю [7], в якій побудована теоретико-імовірнісна модель напівтонового зображення і застосований метод ідентифікації особистості по фотографії особи на основі оптимального байєсівського правила. У іншій роботі [6] вирішувалося завдання автоматичного розпізнавання на основі принципу мінімуму інформаційної неузгодженості. У роботі [5] запропонований оригінальний алгоритм розпізнавання осіб в режимі реального часу. У задачах ідентифікації осіб часто вдаються до використання інваріантних моментів в якості ознак. Так, в роботі застосовується вектор особливостей, що складається з 11 різних моментів, а в роботі [2] досліджуються властивості інваріантних моментів. Показано, що інваріанти мають різну чутливість до змін вхідних даних. відмічу капітальну в цій області працю, в якій систематизуються знання про застосування інваріантних моментів. Також цікава робота [4], присвячена обробці тривимірних зображень, однак завдання 3D-розпізнавання осіб ще недостатньо досліджене. Розуміється, тут не можна не відзначити підходи, пов'язані з виділенням таких особливостей обличчя, як губи, ніс, овал або профіль особи, але з огляду на множини чинників, які ускладнюють аналіз зображень (зашумлення, повороти особи, вираження різних емоцій і ін.) поки немає підходу, що гарантує точне рішення проблеми.

Пропонується комбінований підхід, в якому поєднуються такі інструментальні засоби: методи виділення інваріантних моментів; методи

формування еталонних класів осіб; метрика Евкліда-Махаланобіса [3] і апарат штучних нейронних мереж (ШНМ).

Проведення обчислювальних експериментів показало перспективність комплексного підходу розпізнавання осіб за напівтоновими фотографіями на основі методу інваріантних моментів і класифікаторів на основі метрики Евкліда-Махаланобіса і ймовірнісної нейронної мережі. При використанні метрики Евкліда-Махаланобіса система справляється з поворотами і / або нахилами голови, а також зі змінами яскравості зображення. Імовірнісна нейронна мережа краще справляється з такими «Складними» для системи факторами як закриті очі, змінена міміка особи (посмішка, гримаса і т.п.). Недоліком нейронних мереж являються суттєві часові витрати на навчання при великих обсягах даних. Докорінного поліпшення результату, очевидно, слід очікувати після переходу на 3D технологію розпізнавання осіб і застосування високопродуктивних обчислень для досягнення реального часу при повному циклі обробки зображень.

Для більшості сучасних систем автоматичного розпізнавання осіб основним завданням є завдання порівняння заданого зображення особи з набором зображень осіб з бази даних. Характеристики систем автоматичного розпізнавання осіб в цьому випадку оцінюються шляхом визначення ймовірностей помилкового відмови в розпізнаванні (помилки першого роду) і помилкового розпізнавання (помилки другого роду).

На додаток до можливості помилок для оцінки системи автоматичного розпізнавання осіб часто використовується оцінка стійкості до обурення зображень, що викликається комбінацією зі складними фонами, мінливістю освітлення, змінами зачіски, і т. д.

З огляду на вищевикладене, представляється, що перспективним може бути створення гібридних методів, які використовують переваги і нівелюють недоліки розглянутих вище різних окремих підходів.

1.3.1 Розпізнавання облич на основі метода Віоли-Джонса

Розпізнавання осіб є однією з найбільш вивчених задач в таких областях як цифрова обробка зображень, комп'ютерний зір, біометрія, організація відеоконференцій, створення інтелектуальних систем безпеки та контролю доступу та інші. Процес розпізнавання осіб зазвичай складається з двох етапів: пошук області обличчя на зображенні, і порівняння знайденої особи з особами, які перебувають у базі даних. У даний час метод Віоли-Джонса є найпопулярнішим методом для пошуку області обличчя на зображенні через його високу швидкість і ефективність. Детектор обличчя Віоли-Джонса заснований на основних ідеях: інтегральному перед представленні зображення, методі побудови класифікатора на основі алгоритму адаптивного бустинга (AdaBoost), і методі комбінування класифікаторів у каскадну структуру. Ці ідеї дозволяють побудувати детектор обличчя, здатний працювати у режимі реального часу. Метод головних компонент і вейвлет–перетворення використовують для отримання характеристик зображення. У задачі розпізнавання осіб вони успішно використовуються для порівняння компонент, що характеризують кольорові зображення, з компонентами, що описують невідомі зображення. Метою роботи є створення нового алгоритму, заснованого на комбінації методу Віоли-Джонса, вейвлет-перетворень і методу головних компонент (МГК) для розпізнавання осіб на цифрових зображеннях і відеопослідовностях у режимі реального часу.

Метод був розроблений і представлений в 2001 р Полом Віолою і Майклом Джонсом. Він до сих пір є ефективним для пошуку об'єктів на зображеннях і відеопослідовностях у режимі реального часу [2, 3]. Слід зазначити, що метод має дуже низьку імовірність хибного виявлення особи. Добре працює і виявляє риси обличчя навіть при утриманні об'єкта під невеликим кутом, до 30°. При більшому за 30° куті ймовірність виявлення особи стрімко падає. Така особливість перешкоджає детектуванню обличчя людини, розміщеного під довільним кутом.

Це ускладнює реалізацію такого алгоритму на сучасних системах виробництва зважаючи на ріст потреб.

Щоб вирахувати рівень яскравості ділянки зображення, використовують інтегральні уявлення [4]. Такі уявлення використовуються іншими методами, наприклад, методами вейвлет–перетворень, також у «Speeded up robust feature» (SURF), фільтрами Хаара і ще багатьма розробленими алгоритмами. Інтегральні уявлення дають можливість розрахувати загальну яскравість довільного прямокутника на зображенні. Час розрахунку не буде залежати від площі прямокутника.

Інтегральне представлення зображення являє собою матрицю, що збігається за розмірами з вихідним зображенням. У кожному її елементі зберігається сума інтенсивностей усіх пікселів, що знаходяться лівіше і вище даного елемента.

Вейвлет-перетворення часто застосовується для аналізу нестабільних процесів. Такий засіб показав свою ефективність для застосування до широкого класу задач, пов'язаних з обробкою зображень. Коефіцієнти вейвлет-перетворення тримають інформацію про аналізований процес і використовуваний вейвлет. Тому вибір вейвлета для аналізу визначається тим, яку інформацію необхідно витягти з процесу. Кожен вейвлет має характерні особливості під час прямолінійного руху тимчасової і частотних областях.

Виділення контурів. Було проведено серію експериментів із виділення контурів обличчя, використовуючи оператори бібліотеки Computer Vision Toolbox у середовищі системи комп’ютерної математики – MatLab.

Дана бібліотека дозволяє виконувати виявлення і відстеження об'єктів, детектувати і витягувати ознаки, виконувати їх зіставлення.

У результаті проведених експериментів:

– були визначені межі застосування методів – ефективними методами є методи Собеля та Превітта (рис. 1.3);

– якість виявлення облич – 75%;

– кут нахилу – 14–26%.

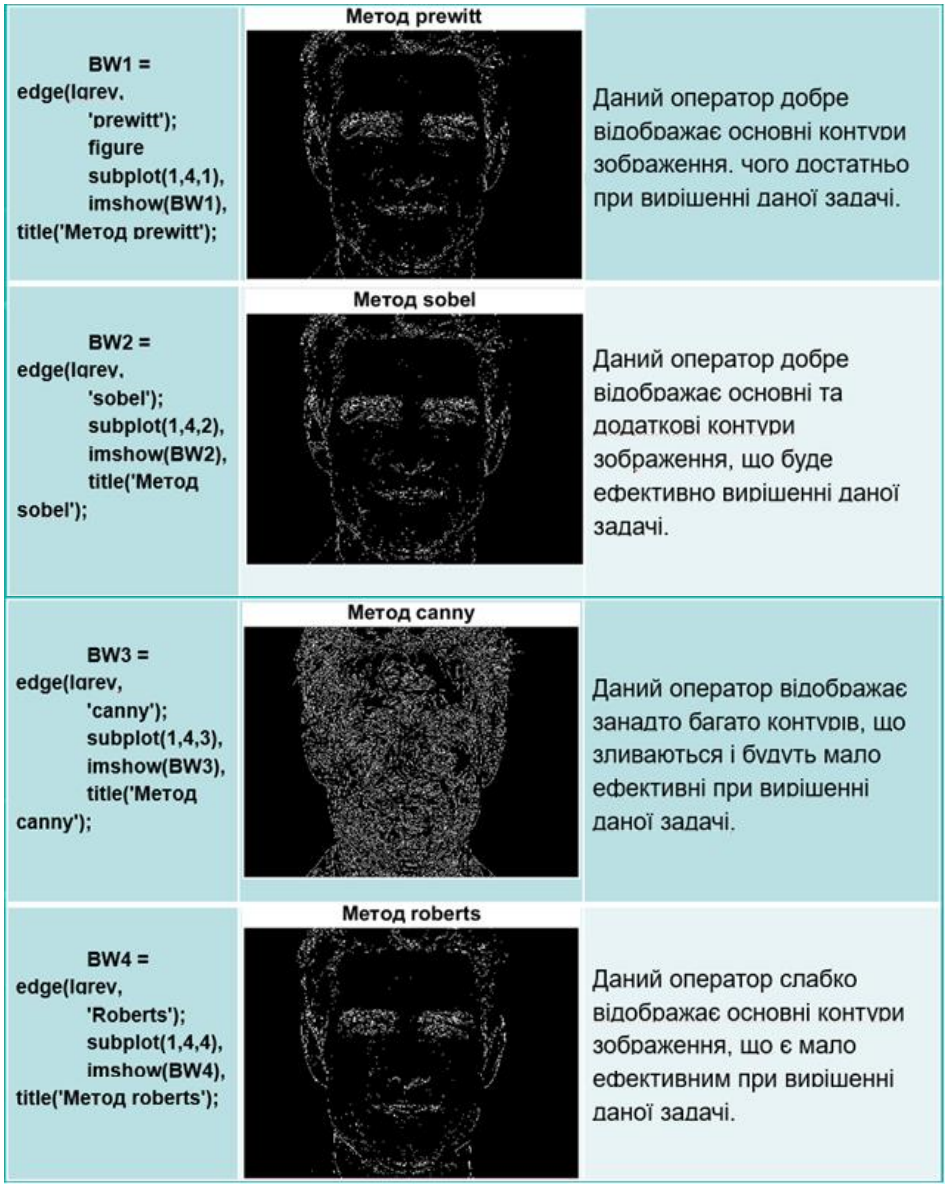


Рисунок 1.3 – Методи виділення контурів

Дані експерименти показали, що найбільш ефективними методами виділення меж є методи Собеля та Превітта. Вони відрізняються від інших методів тим, що добре відображають основні та додаткові контури зображення, не перенасичуючи відображенням шумів на зображеннях. У подальшому будуть використані саме ці методи [28].

Бібліотека комп’ютерного зору OpenCV. Система, що розробляється є програмою, для реалізації якої було обрано мову програмування Python і відповідну бібліотеку OpenCV Python.

OpenCV – це бібліотека алгоритмів для обробки зображень та алгоритмів загального призначення з відкритим кодом і може вільно використовуватися в академічних і комерційних цілях.

Це міжплатформна бібліотека, за допомогою якої можливо розробляти програми «комп’ютерного зору» в режимі реального часу. Основна увага приділяється обробці зображень, захопленню та аналізу відео, включаючи такі функції, як виявлення обличчя та виявлення об’єктів.

Комп’ютерне бачення можна визначити як дисципліну, яка пояснює, як реконструювати, виокремлювати та розуміти 3D простір з його 2D зображень, з точки зору властивостей структури, присутньої у просторі. Він займається моделюванням та реплікацією людського зору за допомогою комп'ютерного програмного та апаратного забезпечення [8].

Комп’ютерне бачення – це побудова явних, змістовних описів фізичних об’єктів за їх зображенням. Результатом комп’ютерного зору є опис або інтерпретація структур у тривимірному просторі.

Набір інструментів Dlib. Dlib – це сучасний набір інструментів, що містить алгоритми машинного навчання та інструменти для створення складного програмного забезпечення на Pyton для вирішення реальних проблем. Він використовується як у промисловості, так і в наукових колах у широкому діапазоні, включаючи робототехніку, вбудовані пристрої, мобільні телефони та великі високопродуктивні обчислювальні середовища. Ліцензування відкритого коду Dlib дозволяє використовувати його в будь-якому додатку безкоштовно [9].

Основні особливості:

– документація (наведено багато прикладних програм);

– високоякісний портативний код (для використання бібліотеки не

потрібні інші пакети, потрібні лише API, які надаються стандартною ОС; увесь код, специфічний для операційної системи, ізольований всередині шарів абстракції ОС, які є якомога меншими, решта бібліотеки або наноситься поверх шарів абстракції ОС);

- алгоритми машинного навчання;
- числові алгоритми;
- алгоритми виведення графічної моделі;
- обробка зображень;
- багатопотоковість;
- графічний інтерфейс користувача;
- алгоритми стиснення даних та цілісності;
- тестування;
- загальні утиліти.

Бібліотека face recognition. За допомогою даної найпростішої у світі бібліотеки розпізнавання облич, можна розпізнавати та обробляти обличчя з у середовищах розробки на мові Python або з командного рядка.

Модель, побудована за допомогою dlib з використанням алгоритмів машинного навчання, має велику точність у порівнянні з іншими аналогами.

Дана бібліотека також забезпечує простий інструмент командного рядка, який дозволяє розпізнавати обличчя в папці зображень.

Бібліотека використовує багато вбудованих бібліотек, таких як Dlib, і використовує машинне навчання, щоб розпізнавати обличчя із точністю 99,38%.

1.3.2 Штучні нейронні мережі

В даній роботі метою є розробка нейромережевої системи розпізнавання рис обличчя. Для цього спочатку потрібно вивчити методи використання

нейронних мереж для розпізнавання рис обличчя. Це потребує визначення перспективних та актуальних нейромережевих архітектур. Після вибору архітектури нейронної мережі потрібно детально вивчити особливості використання мереж даної архітектури.

Після цього потрібно сформувати алгоритм визначення вхідних даних нейронної мережі, тобто розмірність, метод чисельного подання вхідних даних та ін.

Розробка алгоритму навчання та використання нейронної мережі також є важливим кроком у розробці системи.

Проведений аналіз сучасного стану нейромережевих технологій дозволяє сформулювати висновок про те, що доцільність застосування конкретного типу НМ слід визначати на основі співставлення характеристик мережі з умовами прикладної задачі. До вказаних характеристик та умов відносяться:

– параметри навчальних даних;
– загальні обмеження процесу навчання;
– вимоги до обчислювальних потужностей;
– вимоги до вихідної інформації;
– обмеження технічної реалізації НМ;
– сфера застосування.

Штучний нейрон є базовим модулем нейронних мереж. Він моделює основні функції природного нейрона (див. рис. 1.4).

При функціонуванні нейрон одночасно отримує багато вхідних сигналів. Кожен вхід має свою власну синаптичну вагу, яка надає входу вплив, необхідний для функції суматора елемента обробки. Ваги є мірою сили вхідних зв'язків і моделюють різноманітні синаптичні сили біологічних нейронів.

Вхідні сигнали x_n зважені ваговими коефіцієнтами з'єднання w_n додаються, проходять через передатну функцію, генерують результат і виводяться.

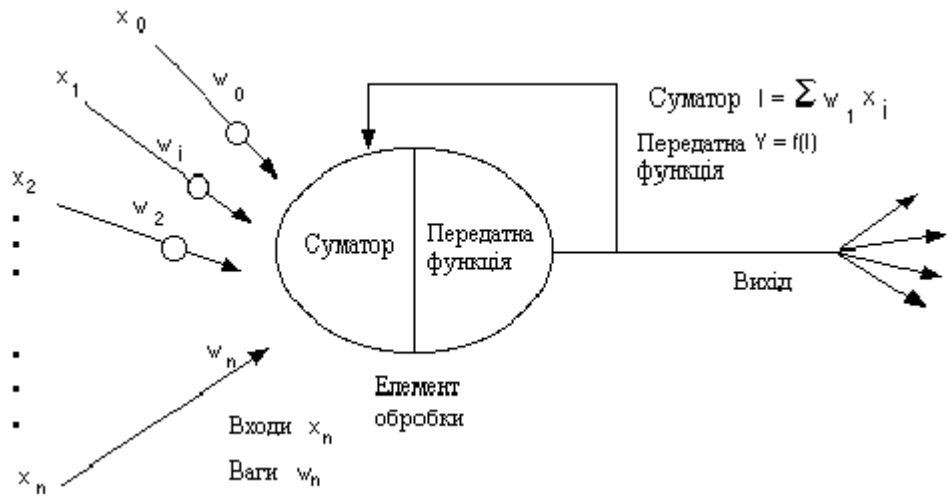


Рисунок 1.4 – Базовий штучний нейрон

В програмних реалізаціях штучні нейрони називають «елементами обробки» або «процесорами» і вкладають в них більше можливостей, ніж в базовому штучному нейроні, що описаний вище.

На рисунку 1.5 зображена детальна схема штучного нейрону.

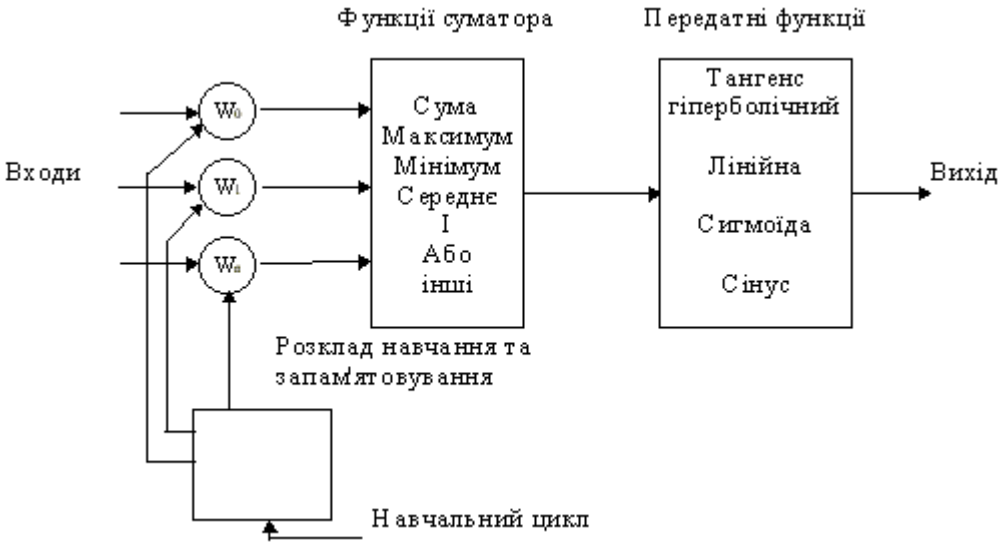


Рисунок 1.5 – Модель «елементу обробки»

Функція суматора може бути складнішою, наприклад, вибір мінімуму, максимуму, середнього арифметичного, добутку або обчислюватися за іншим

алгоритмом.

Перед надходженням до передатної функції вхідні сигнали та вагові коефіцієнти можуть комбінуватись багатьма способами.

Результат функції суматора перетворюється у вихідний сигнал через передатну функцію. В передатній функції для визначення виходу нейрона загальна сума порівнюється з деяким порогом (зазвичай, це діапазон [0, 1] або [-1,1] або інше) за допомогою певного алгоритму.

Штучні нейромережі конструюються з базового блоку – штучного нейрону. Іншою властивістю нейромереж є величезна кількість зв'язків, які пов'язують окремі нейрони.

За архітектурою зв'язків, більшість відомих нейромереж можна згрупувати у два великих класи:

а) мережі прямого поширення (з односкерованими послідовними зв'язками);

б) мережі зворотного поширення (з рекурентними зв'язками).

Мережі прямого поширення відносять до статичних, тут на входи нейронів надходять вхідні сигнали, які не залежать від попереднього стану мережі.

Рекурентні мережі вважаються динамічними, оскільки за рахунок зворотних зв'язків (петель) входи нейронів модифікуються в часі, що призводить до зміни станів мережі [8].

Оригінальність нейромереж, аналогічно біологічному мозку, полягає у здатності до навчання за певними прикладами, що складають навчальну множину. Процес навчання нейромереж полягає у налаштуванні архітектури та вагових коефіцієнтів синаптичних зв'язків у відповідності до даних навчальної множини для ефективного вирішення поставленої задачі.

Для навчання нейромереж можливо у випадку:

– навчання з вчителем (контрольоване навчання);

– навчання без вчителя (неконтрольоване навчання).

У більшості реалізацій нейромереж використовується контрольоване навчання, де біжучий вихід постійно порівнюється з бажаним виходом. Вагові коефіцієнти зв’язків на початку встановлюються випадковим чином (ініціалізація мережі), та під час наступних ітерацій коректуються, щоб досягти найбільшої відповідності між бажаними та біжучим виходами.

Перед використанням, нейромережа з контрольованим навчанням повинна пройти навчання. Фаза навчання займає деякий час. Навчання вважається закінченим при досягненні нейромережею визначеного користувачем рівня ефективності і запланованої статистичної точності [2].

Якщо після контрольованого навчання нейромережа ефективно опрацьовує дані навчальної множини, важливим стає її ефективність при роботі з даними, які не використовувались для навчання. У випадку отримання незадовільних результатів для тестової множини, навчання продовжується.

Неконтрольоване навчання може бути великим надбанням у майбутньому. Воно проголошує, що комп’ютери можуть самонавчатись у справжньому роботизованому сенсі. На даний час, неконтрольоване навчання використовується в мережах відомих, як самоорганізовані карти (selforganizingmaps). Мережі не використовують зовнішніх впливів для коректування своїх ваг і внутрішньо контролюють свою ефективність, шукаючи регулярність або тенденції у вхідних сигналах та здійснюють адаптацію відповідно до навчальної функції. Навіть без повідомлення правильності чи неправильності дій, мережа повинна мати інформацію відносно власної організації, яка закладена у топологію мережі та навчальні правила.

Побудуємо математичну модель описаного процесу (див. рис. 1.6).

На рисунку зображена модель нейрона з трьома входами (дендритами), причому синапси цих дендритів мають ваги w_1, w_2, w_3. Нехай до синапсів надходять імпульси сили x_1, x_2, x_3 відповідно, тоді після проходження синапсів і дендритів до нейрона надходять імпульси $w_1 x_1$, $w_2 x_2$, $w_3 x_3$.

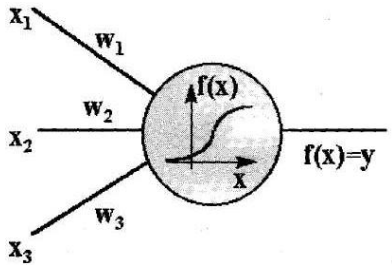


Рисунок 1.6 – Спрощена модель нейрона

Нейрон перетворить отриманий сумарний імпульс

$$x= w_1 x_1+ w_2 x_2+ w_3 x_3$$

відповідно до деякої передатної функції *f(x)*. Сила вихідного імпульсу дорівнює

$$y=f(x)=f(w_1x_2+ w_2 x_2+ w_3 x_3).$$

Таким чином, нейрон цілком описується своїми вагами w_k і передатною функцією *f(x)*. Одержавши набір чисел (вектор) w_k як входи, нейрон видає деяке число *y* на виході.

Розглянемо узагальнену модель нейрона (рис. 1.7), зв’язану з першими спробами формалізувати опис функціонування нервової клітки.

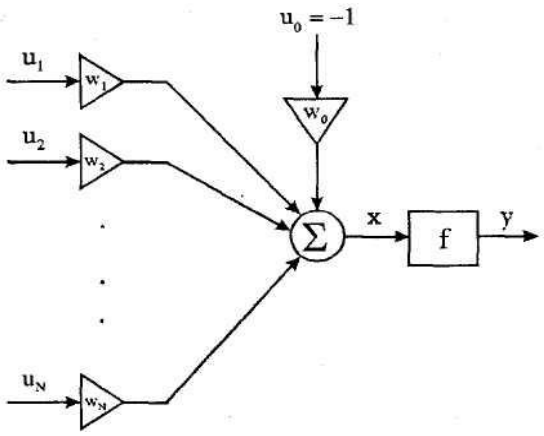


Рисунок 1.7 – Узагальнена модель нейрона

Введемо наступні позначення:

$u_1, ..., u_N$ – вхідні сигнали даного нейрона, що приходять від інших нейронів;

$w_1.....w_N$ – синаптичні ваги;

y – вихідний сигнал нейрона;

v – граничне значення.

Формула, що описує функціонування нейрона, має вигляд

$$y = \begin{cases} 1 \quad при \quad \sum_{i=1}^{N} w_1 u_1 \geq v, \\ 0 \quad при \quad \sum_{i=1}^{N} w_1 u_1 < v. \end{cases} \tag{1.1}$$

Модель (1.1) може бути представлена у виді

$$y = f\left(\sum_{i=0}^{N} w_i u_i\right), \tag{1.2}$$

де

$$f(x) = \begin{cases} 1 \quad при \quad x \geq 0, \\ 0 \quad при \quad x < 0. \end{cases} \tag{1.3}$$

А також $w_0 = v$, $u_0 = 1$.

Формула (1.1) описує модель нейрона, представлену на рис. 1.7. Ця модель була запропонована в 1943 р. Маккаллоком і Питтсом. В якості функції f може прийматися не тільки одинична функція (1.3), але й інші граничні функції виду

$$f(x) = \begin{cases} 1 \quad при \quad x \geq 0, \\ -1 \quad при \quad x < 0. \end{cases} \tag{1.4}$$

Або

$$f(x)=\begin{cases} 1 & при \quad x>1, \\ -1 & при \quad x<-1, \\ x & при \quad |x|\le 1. \end{cases} \tag{1.5}$$

На початковій фазі моделювання біологічних нейронних мереж застосовувалися граничні функції. В даний час найчастіше використовується сігмоїдальна функція, яка обумовлена виразом

$$f(x)=\frac{1}{1+e^{-\beta x}}>0. \tag{1.6}$$

Відзначимо, що при $\beta\to\infty$ характеристика (1.6) прагне до граничної уніполярної функції (1.3). Як альтернативу застосовується функція гіперболічного тангенса

$$f(x)=th\left(\frac{\alpha x}{2}\right)=\frac{1-e^{-\alpha x}}{1+e^{-\alpha x}}>0. \tag{1.7}$$

У цьому випадку характеристика (1.7) прагне до граничної біполярної функції (1.4) при $\alpha\to\infty$. Приклади функції f у моделі (1.2) показані на рисунку 1.8.

В загальному випадку задача навчання ШНМ зводиться до знаходження деякої функціональної залежності $Y=F(X)$ де X – вхідний вектор, а Y – вихідний вектор. В загальному випадку така задача, при обмеженому наборі вхідних даних має нескінченну множину розв’язків.

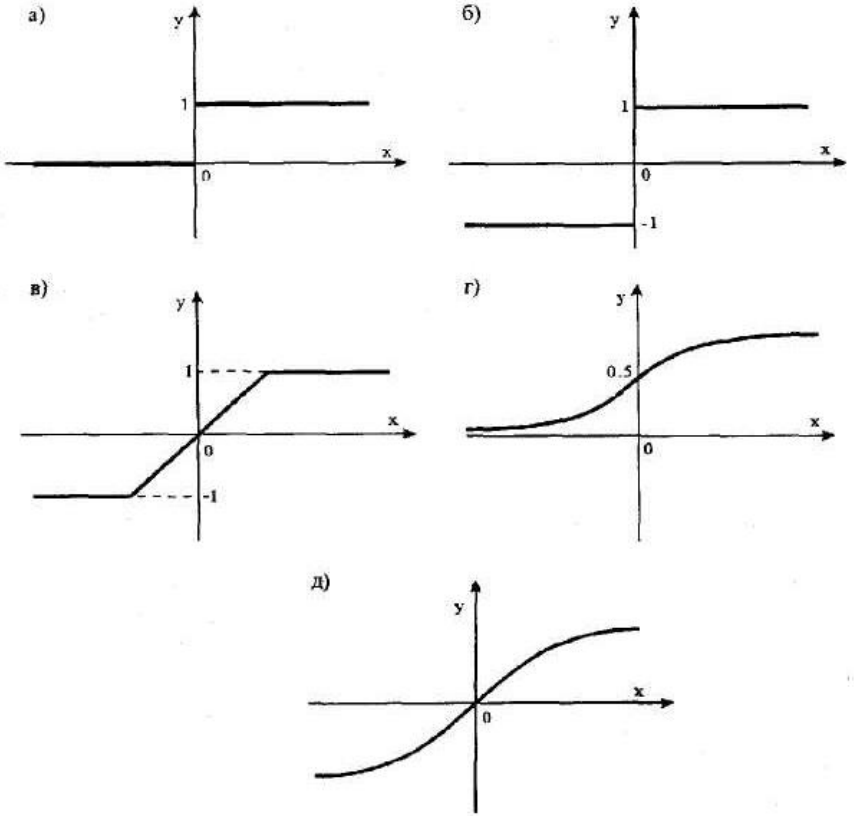


Рисунок 1.8 – Приклади функції *f*

Для обмеження простору пошуку при навчанні ставиться задача мінімізації цільової функції похибки ШНМ, що знаходиться за методом найменших квадратів:

$$E(w) = \frac{1}{2}\sum_{j=1}^{p}(y_j - d_j) \tag{1.8}$$

де y_j – значення *j-го* виходу нейронної мережі, d_j – цільове значення *j-го* виходу, *p* – кількість нейронів в вихідному шарі.

$$\Delta w = -\eta \cdot \frac{\partial E}{\partial w_{ij}} \tag{1.9}$$

де η – параметр, що визначає швидкість навчання.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{dS_j} \cdot \frac{\partial S_j}{\partial w_{ij}} \tag{1.10}$$

де y_j – значення *j-го* виходу нейрону, S_j – зважена сума вхідних сигналів, що визначається за формулою (1.1), при цьому множник

$$\frac{\partial S_j}{\partial w_{ij}} \equiv x_i \tag{1.11}$$

де x_i – значення *i-го* входу нейрону.

Далі розглянемо визначення першого множника формули (1.7)

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{dS_k} \cdot \frac{\partial S_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{dS_k} \cdot w_{jk}^{(n+1)} \tag{1.12}$$

де k – кількість нейронів в шарі $n+1$.

Тепер розглянемо повний алгоритм навчання нейронної мережі:

а) подати на вхід ШНМ один з потрібних зразків і визначити значення виходів нейронів ШНМ;

б) розрахувати $\delta_j^{(N)}$ для вихідного шару ШНМ за формулою (1.12) і визначити зміни ваг $\Delta w_{ij}^{(N)}$ вихідного шару N за формулою (1.13);

в) розрахувати за формулами (1.11) і (1.13) відповідно $\delta_j^{(N)}$ і $\Delta w_{ij}^{(N)}$ для

інших шарів ШНМ, N=N-1..1;

г) скоректувати всі ваги ШНМ;

д) $$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t);$$ (1.13)

е) якщо помилка значна, то перейти на крок 1.

На другому етапі навчання мережі почергово в довільному порядку вибираються вектори з навчальної послідовності [6].

Дослідження складових та етапів побудови нейромережі. Незалежно від розташування та функціонального призначення, всі штучні нейронні елементи мають спільні компоненти. Розглянемо сім основних компонент штучного нейрона.

Компонента 1. Вагові коефіцієнти. При функціонуванні нейрон одночасно отримує багато вхідних сигналів. Кожен вхід має свою власну синаптичну вагу, яка надає входу вплив, необхідний для функції суматора елемента обробки.

Компонента 2. Функція суматора. Першим кроком дії нейрону є обчислення зваженої суми всіх входів. Математично, вхідні сигнали та відповідні їм ваги представлені векторами (x_{10}, x_{20} ... x_{n0}) та (w_{10}, w_{20} . . . w_{n0}). Добуток цих векторів є загальним вхідним сигналом.

Компонента 3. Передатна функція. Результат функції суматора є зваженою сумою вхідних сигналів, що перетворюється у вихідний сигнал через алгоритмічний процес відомий як передатна функція.

Компонента 4. Масштабування. Після передатної функції вихідний сигнал проходить додаткову обробку масштабування, тобто результат передатної функції множиться на масштабуючий коефіцієнт і додається зміщення.

Компонента 5. Вихідна функція (змагання). По аналогії з біологічним нейроном, кожний штучний нейрон має один вихідний сигнал, який передається до сотень інших нейронів.

Компонента 6. Функція похибки та поширюване назад значення. У

більшості мереж, що застосовують контрольоване навчання обчислюється різниця між спродукованим та бажаним виходом. Похибка відхилення (біжуча похибка) перетворюється функцією похибки відповідно до заданої мережної архітектури. В базових архітектурах похибка відхилення використовується безпосередньо, в деяких парадигмах використовується квадрат або куб похибки зі збереженням знаку.

Компонента 7. Функція навчання. Метою функції навчання є налаштування змінних ваг з'єднань на входах кожного елемента обробки відповідно до певного алгоритму навчання для досягнення бажаного результату. Існує два типи навчання: контрольоване та неконтрольоване [5].

1.3.3 Опис нейронної мережі глибокого навчання

Рішення про приналежність вхідного зображення класу зображень облич приймається на основі порівняння величини подібності зображення з обличчями і порога на результат такого зіставлення. Використовується функція подібності у вигляді косинуса кута між нормованими векторами, такими, що їх середні дорівнюють нулю:

$$\begin{cases} S_0(\vec{I}_1, \vec{I}_2) = \dfrac{\left(\vec{I}_1, \vec{I}_2\right)}{\left|\vec{I}_1\right\|\vec{I}_2\right|}, \\ \dfrac{1}{N}\sum\limits_{i=0}^{N-1} I_1^i = 0, \\ \dfrac{1}{N}\sum\limits_{i=0}^{N-1} I_2^i = 0, \end{cases} \tag{1.14}$$

де $S_0(\vec{I}_1, \vec{I}_2)$ – функція подібності, $\vec{I}_1$ та $\vec{I}_2$ – порівнювані зображення,

представлені у векторній формі, N – розмірність векторів.

Функція подібності зображень (1) інваріантна до рівномірних змін яскравості і контрасту зображень, які обумовлені відмінностями в умовах одержання зображень і апаратурі вводу. Для виявлення облич різних масштабів будується піраміда зображень по масштабах, і здійснюється пошук по всіх рівнях піраміди. Завдяки цьому, метод дозволяє не тільки знаходити положення облич на зображенні, але і визначати їх масштаб, який пов'язаний з рівнем піраміди, що дав найкращу відповідність. Для підвищення швидкості роботи і інваріантності порівняння до індивідуальних особливостей, метод застосовується до зменшених зображень облич з горизонтальним розміром шаблона – 12 точок. Для зменшення впливу частин зображень облич, підданих найбільш частим змінам – зачіска, навколишній фон, для розрахунку шаблону використовуються зображення центральної області обличчя, що включає брови, очі, ніс і рот. Якість описаного методу визначається сумарною помилкою поділу двох класів зображень: облич і фону. Для оцінки такої помилки використовуються щільності розподілу результату кореляції шаблона обличчя з зображеннями облич з наявної бази даних і зображеннями фону.

Для представлення властивостей облич при рішенні задачі ідентифікації використовуються коефіцієнти згорток вихідного зображення з функціями Габора різних масштабів і кутів повороту. Функції Габора локалізовані в просторовій і частотній області і мають вигляд плоскої хвилі з хвильовим вектором $\vec{k}$, на яку накладена гаусовська огинаюча функція шириною σ/k, де $\sigma = 2\pi$:

$$\psi_j(\vec{x}) = \frac{k_j^2}{\sigma^2}\exp\left(-\frac{k_j^2 x^2}{2\sigma^2}\right)\left[\exp\left(i\vec{k}_j\vec{x}\right) - \exp\left(-\frac{\sigma^2}{2}\right)\right], \qquad (1.15)$$

де

$$\vec{x} = (x, y), \vec{k}_j = \left(k_{jx}, k_{jy}\right), k_j^2 = \left|\vec{k}_j\right|^2, x^2 = |\vec{x}|^2, \vec{k}_j\vec{x} = k_{jx}x + k_{jy}y. \quad (1.16)$$

Нормувальний коефіцієнт, друга експонента в квадратних дужках, отриманий з умови рівності нулю інтеграла від функцій Габора по всій області визначення, що дає інваріантність згортки довільного зображення з функціями Габора щодо постійного зсуву зображення по шкалі яскравості. У даній роботі використовувалися функції Габора п'яти різних масштабів, $\nu = \{0,...,4\}$, і восьми кутів повороту, $\mu = \{0,...,7\}$. Кожна функція визначалася характеристичним хвильовим вектором:

$$\vec{k}_j = \begin{pmatrix} k_\nu \cos\phi_\mu \\ k_\nu \sin\phi_\mu \end{pmatrix}, \quad k_\nu = 2^{-\frac{\nu+2}{2}}\pi, \quad \phi_\mu = \mu\frac{\pi}{8}, \quad (1.17)$$

де індекс $j = \mu + 8\nu$. Вибір саме такого набору функцій Габора обумовлений найкращою апроксимацією габоровськимивейвлетами вихідної області зображення. Повне хвильове перетворення

$$J_j(x, y) = \int\int I(x', y')\psi_j(x - x', y - y')dx'dy' \quad (1.18)$$

дає 40 комплексних коефіцієнтів $J_j(x, y)$ у кожній точці зображення $I(x, y)$ (5 масштабів і 8 кутів). Операція обчислення згортки (1.18) має значну складність, тому для прискорення розрахунків використовувалося ШПФ.

Для порівняння векторів ознак, отриманих на основі (1.18), як і при виявленні облич, використовується функція у вигляді косинуса кута між ними (1). Така функція порівняння інваріантна до рівномірних змін яскравості і контрастності зображень, на основі яких вектора ознак були отримані. Згідно

методу еластичного зіставлення графів, зображення обличчя трансформується в граф обличчя, що містить його властивості. Конфігурація графа визначає геометричні особливості обличчя, вектори ознак у вузлах – локальні властивості зображення. У даній роботі використовується конфігурація графа обличчя у виді прямокутної сітки, при цьому перша зверху лінія вузлів графа відповідає лінії брів; друга – лінії очей; четверта – лінії, що проходить через кінчик носу; шоста – лінії губ. Граф розташовується в центральній частині обличчя, що дозволяє виключити вплив на формування ознак зачіски і навколишнього фону.

Для визначення близькості зображень облич, порівнюються відповідні їм графи облич за допомогою функції порівняння:

$$S_G(G^1, G^2) = \frac{1}{N}\sum_{n=0}^{N-1} S_0(G_n^1, G_n^2), \qquad (1.19)$$

де S_G – результат порівняння графів; N – число вузлів у графі; n – індекс, що визначає номер вузла; G_n^1, G_n^2 – вектори ознак, що відповідають n-му вузлу графу; $S_0(G_n^1, G_n^2)$ – функція порівняння векторів ознак (1.19). Рішення про належність зображення класу зображень облич даної людини здійснюється за допомогою порогу на результат порівняння графу зображення, що тестується, і графу обличчя відомої людини, що міститься в базі даних. При аналізі зображення обличчя, конфігурація графу підстроюється під його пропорції для того, щоб домогтися відповідності між порівнюваними точками зображень облич.

Для об’єднання етапів пошуку обличчя та розпізнавання в єдиний комплекс запропоновано підхід, який враховує вплив відхилень у визначенні масштабів, що находяться під час пошуку, на результат ідентифікації. Для зменшення цього впливу запропоновано визначати шаг дискретизації по масштабах при побудові

піраміди зображень в методі пошуку на базі заданого обмеження на помилку розпізнавання.

Способи оцінки якості методу розпізнавання залежать від задачі, для вирішення якої його передбачається використовувати. Як правило, алгоритми і методи автоматичної ідентифікації людини за зображенням обличчя розробляються для вирішення задач контролю доступу і пошуку в базі даних. Для оцінки якості методу при вирішенні задачі контролю доступу використовуються щільності розподілу для результатів порівняння зображень облич різних людей і однієї людини. При цьому поріг розпізнавання визначається на підставі важливості для конкретних умов помилок, пов’язаних з неправильною ідентифікацією і відмовленням від розпізнавання. При оцінці якості методу для рішення задачі пошуку в базі даних використовується відсоток знаходження правильної відповідності. При цьому використовуються еквівалентні бази даних, тобто зображення обличчя людини, введеного в тестову базу даних, повинне міститися й у базі даних, у якій здійснюється пошук. При цьому ці зображення не повинні бути ідентичними. Іноді використовується більш складна оцінка якості роботи методу, застосовувана, наприклад, при тестуванні в рамках програми FERET. Методика тестування полягає в розрахунку залежності відсотка перебування правильної відповідності у визначеному числі знайдених відповідностей з найбільшими вагами від цього числа. У даній роботі така методика використовується для порівняння розроблених методів ідентифікації з аналогами. Необхідно зазначити, що об’єктивна оцінка методів автоматичної ідентифікації по зображенню обличчя надзвичайно ускладнена, тому що результат тестування сильно залежить від використовуваної бази даних. Для вирішення цієї задачі необхідно або реалізувати велику кількість вже розроблених підходів, або використовувати для тестування велику розмаїтість баз даних зображень облич, які використовували окремі автори для дослідження роботи своїх алгоритмів. У цих умовах зручно використовувати відносні

показники роботи методів, отримані для використовуваних раніше підходів і розроблених модифікацій на одній базі даних зображень облич. Крім того, у рамках вже згаданої програми FERET було проведене тестування ряду систем розпізнавання на одній базі даних, що дає перевагу використання для порівняльного аналізу розроблених методів з аналогами саме бази даних FERET.

Перша модифікація, полягає в розбивці зображень на окремі ділянки і порівняння їх вроздріб, що дає інваріантість результату порівняння до локальних змін яскравості і контрастності зображень у зазначених областях. У цьому випадку результат порівняння двох зображень обчислюється за формулою:

$$S_1 = \sum_{i=1}^{N} S_0^i \ , \tag{1.20}$$

де S_0^i – результат кореляції, розрахований по області з індексом {i} на основі формули (1), N – кількість областей. Такий спосіб порівняння дозволяє більш гнучко оцінювати подібність зображень, одержуваних в умовах з різним напрямком освітлення.

Друга модифікація кореляційного методу пошуку області обличчя базується на використанні декількох шаблонів розрахованих методом власних облич. Власні обличчя було обчислено як власні вектори коваріаційної матриці, отриманої на основі навчальної бази даних, що складалася з 376 зображень облич:

$$S_X = \sum_{k=1}^{N} (x_k - \mu)(x_k - \mu)^T \ , \ W_{opt} = \arg\max_W \left(\left| W^T S_X W \right| \right), \tag{1.21}$$

де S_X – матриця коваріації, N – кількість облич у базі даних, μ – середній по базі даних вектор (середнє обличчя), x_k – k-й вектор з бази даних, W_{opt} –

матриця власних векторів (власних облич).

Характер зменшення власних значень матриці показав можливість використання перших 3 власних векторів, що відбивають найбільш істотні зміни в просторі зображень облич. Поділяюча класи зображень облич і фону поверхня побудована на основі мінімізації функції сумарної помилки класифікації градієнтним методом. Передбачається, що поділяюча поверхня є поверхнею другого порядку.

Порівняльна характеристика кореляційного методу пошуку, що був описаний в другому розділі, та запропонованих модифікацій, отримана на одному тестовому наборі зображень облич і фону (використовувалося 517 зображень облич і 10000 зображень фону), наведена в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика розроблених методів виявлення області обличчя на зображенні

Характеристики	Звичайний метод	Метод з поділом шаблона на ділянки	Метод із трьома шаблонами
Сумарна помилка класифікації, %	17.7	8.6	6.3
Помилки пропуску облич, %	10.9	4.9	4.8
Помилкове виявлення, %	6.8	3.7	1.5
Час обробки зображення розміром 320×240 точок на комп'ютері Celeron-350, сек.	0.20-0.25	0.20-0.25	0.4-0.5

При тестуванні здійснювався пошук облич 8 масштабів, горизонтальний розмір яких змінювався від 34 до 91 точок.

Таким чином, запропоновані модифікації кореляційного методу виявлення обличчя дозволили зменшити сумарну помилку класифікації більш ніж у 2 рази.

Швидкість і якість роботи розроблених методів дозволяє використовувати їх у різноманітних системах, що здійснюють автоматичний аналіз зображень облич, як методів попередньої обробки.

В якості методу розпізнавання обличчя на основі 2D зображення, згідно із проаналізованих джерел [2, 6, 7], обрано метод головних компонент (PCA) на основі характеристик продуктивності, розміру моделі та часу навчання [6].

Відповідно до вибраних методів розпізнавання для 2D та 3D зображень, розроблено наступну схему процесу розпізнавання для програмного сервісу розпізнавання облич з використанням 3D сенсора PrimeSenseCarmine 1.08 (див. рис. 1.9).

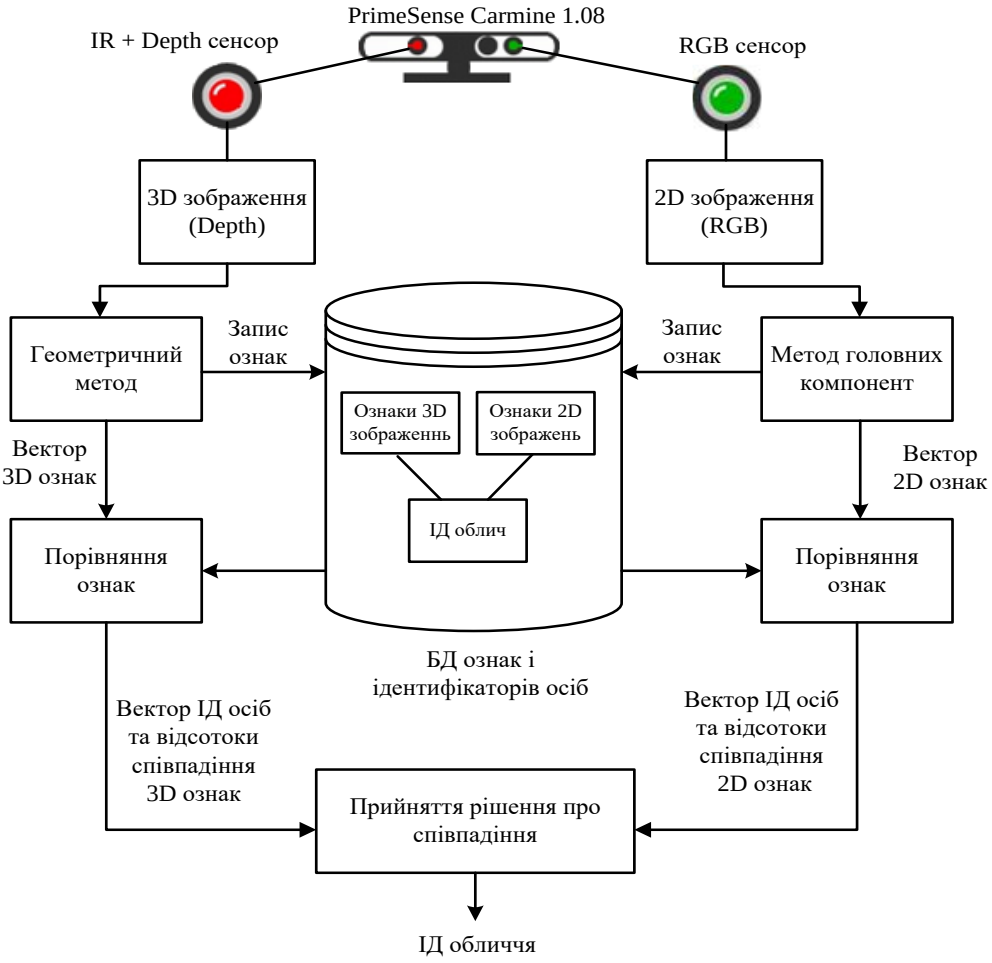


Рисунок 1.9 – Схема процесу розпізнавання обличчя з використанням нейромережі

Основна ідея – паралельний процес розпізнавання обличчя на основі 2D та 3D зображень відповідним алгоритмом. Прийняття рішення в даному випадку відбувається із врахуванням результатів розпізнавання кожного із методів. Даний варіант комбінованого 2D та 3D розпізнавання суттєво збільшує ефективність та точність розпізнавання, збільшуючи при цьому надійність програмного сервісу.

1.4 Висновки до розділу 1

У першому розділі розглянуто основні поняття системи розпізнавання облич, проаналізовано предметну область, досліджено декілька відмінних способів розпізнавання, сфери і продукти у яких ці методи використовуються.

2 ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ІДЕНТИФІКАЦІЇ ОБЛИЧЧЯ ЛЮДИНИ

2.1 Розробка архітектури та структури програмного комплексу

Розпізнавання осіб складається з безлічі взаємозалежних підзадач, в яких потрібно:

- проаналізувати зображення і знайти на ньому все обличчя;
- навчитися розпізнавати кожну особу, навіть якщо воно дивним чином повернуто, або якщо освітлення погане;
- вміти визначати унікальні риси обличчя, які відрізняють одну людину від інших, наприклад, розмір очей, форма обличчя і так далі;
- порівняти виявлені унікальні особливості цієї особи з усіма людьми, яких система вже знає, щоб зрозуміти, хто зображений на фото.

Мозок людини робить все миттєво і непомітно для нас. Можна сказати, що люди настільки звикли до процедури розпізнавання осіб, що бачать обличчя навіть в повсякденних об’єктах (психологічний феномен парейдолії).

Комп’ютери не в змозі до таких узагальнень, тому нам доведеться вчити їх цьому крок за кроком.

Для цього потрібно побудувати свого роду конвеєр, на якому виконується кожен крок розпізнавання по-окремо, а потім передаємо результат поточного кроку наступного. Іншими словами, послідовно з’єднується кілька алгоритмів машинного навчання:

- знаходження обличчя на зображенні;
- аналіз рис обличчя;
- порівняння з вже відомими особами;
- зробити висновки.

Систему реалізовано за допомогою Visual Studio Code. Це кросплатформений редактор вихідного коду, розроблений Microsoft (рис. 2.1). Позиціонується як «легкий» редактор коду для кросплатформеної розробки додатків.

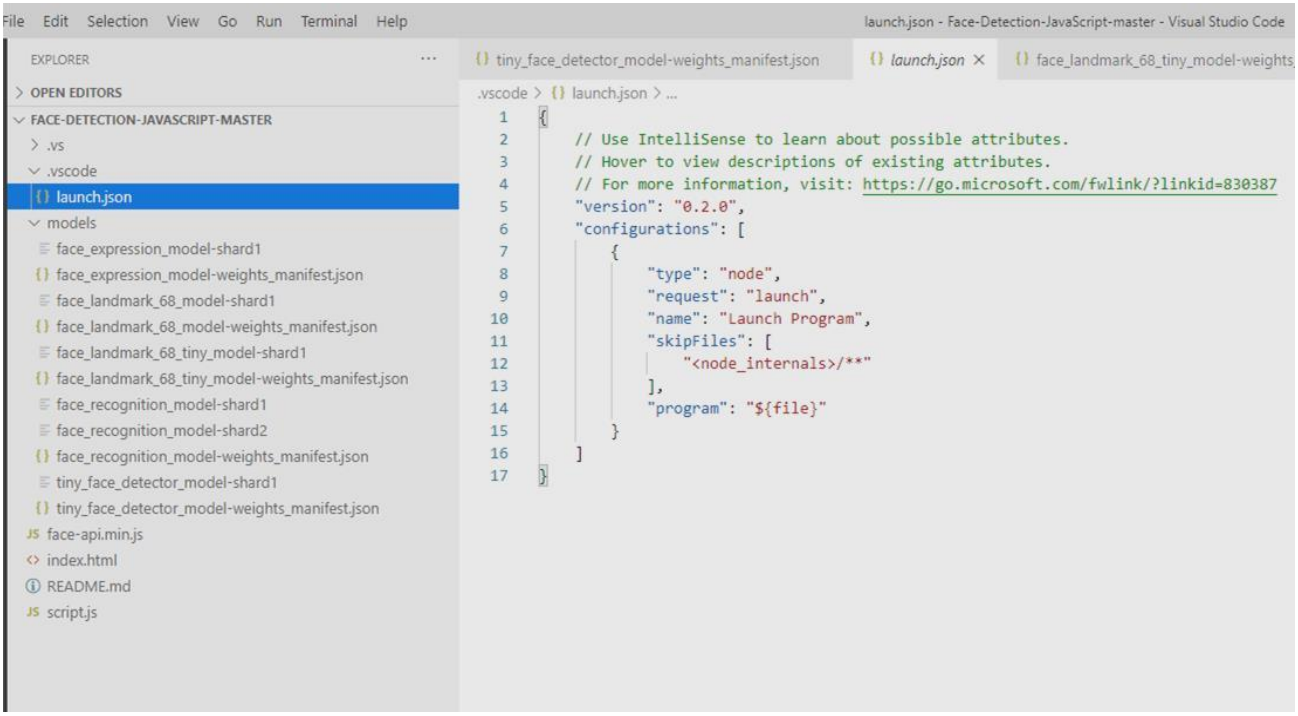


Рисунок 2.1 – Зображення робочої зони Visual Studio Code

Запуск програми здійснюється за допомогою компілятора Go Live (рис. 2.2).

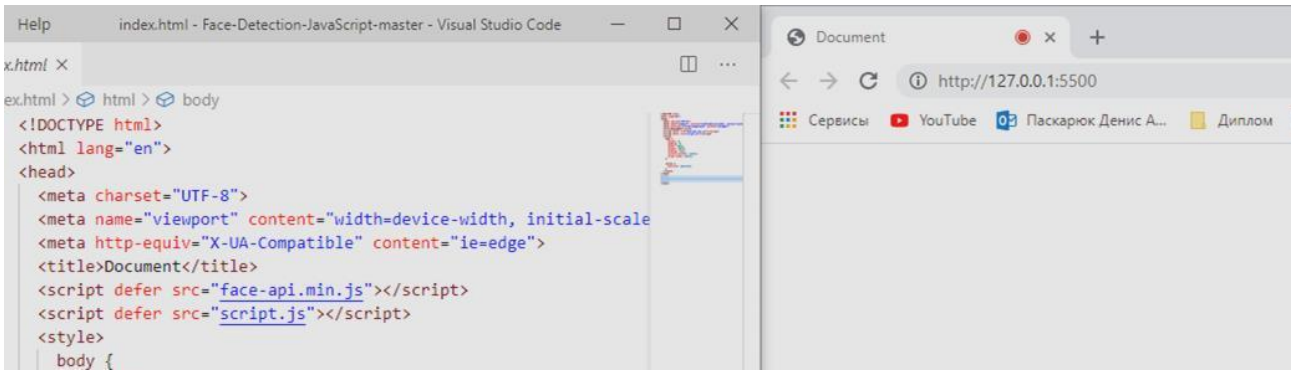


Рисунок 2.2 – Відображення запуску програми ідентифікації

Програмний додаток складається з таких бібілотек:

– .vs;

– .vscode;

– models;

– face-api. min.js;

– script.js.

Бібліотеки .vs та .vscode є стандартними бібліотеками VS Code, та дозволяють робочому простору функціонувати за допомогою стандартних методів.

Бібліотека models зберігає у собі такі методи (рис. 2.3):

– faceapi.nets.tinyFaceDetector.loadFromUri('/models');

– faceapi.nets.faceLandmark68Net.loadFromUri('/models');

– faceapi.nets.faceRecognitionNet.loadFromUri('/models');

– faceapi.nets.faceExpressionNet.loadFromUri('/models').

Бібліотека face-api.min.js зберігає у собі методи опису навчання розпізнавання нейромережею об’єктів у просторі камери.

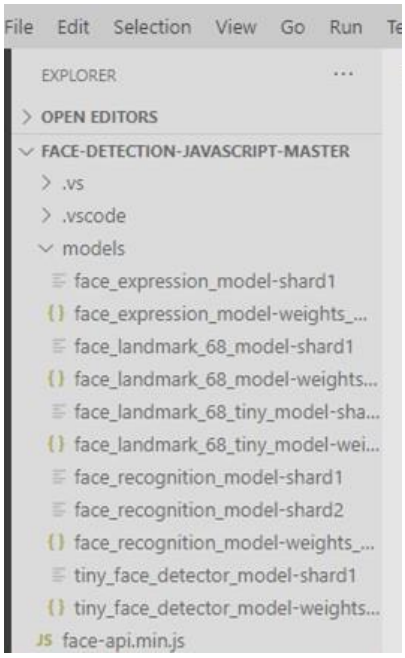


Рисунок 2.3 – Зображення структури додатку

Файл script.js у собі має усі методи, які необхідні при розпізнаванні об’єктів за допомогою нейромережі (див. рис. 2.4).

Метод опорних точки (Face Landmarks) – ідентифікує обличчя за допомогою набору точок, від 5 до 68.

Рисунок 2.4 – Зображення файлу script.js

Face-recognition.js. – метод дозволяє швидко і не надійно визначити осіб на зображенні використовуючи нейромережу глибокого навчання або простий фронтальний розпізнавач.

Метод Promise.all (рис. 2.5) відповідає за ідентифікацію обличчя, по моделям, які вказані у контейнерах Models (див. рис. 2.6).

```
Promise.all([ faceapi.nets.tinyFaceDetector.loadFromUri('/models'),
faceapi.nets.faceLandmark68Net.loadFromUri('/models'),
faceapi.nets.faceRecognitionNet.loadFromUri('/models'),
faceapi.nets.faceExpressionNet.loadFromUri('/models')
]).then(startVideo)
```

Рисунок 2.5

Для встановлення положення обличчя використовується faceapi.nets.faceRecognitionNet.loadFromUri('/models').

Для відстеження положення губ та визначення емоцій використовується faceapi.nets.faceExpressionNet.loadFromUri('/models').

Дані моделі ідентифікації дозволяють розпізнавати не лише одну людину, а усіх присутніх у об’єктиві камери.

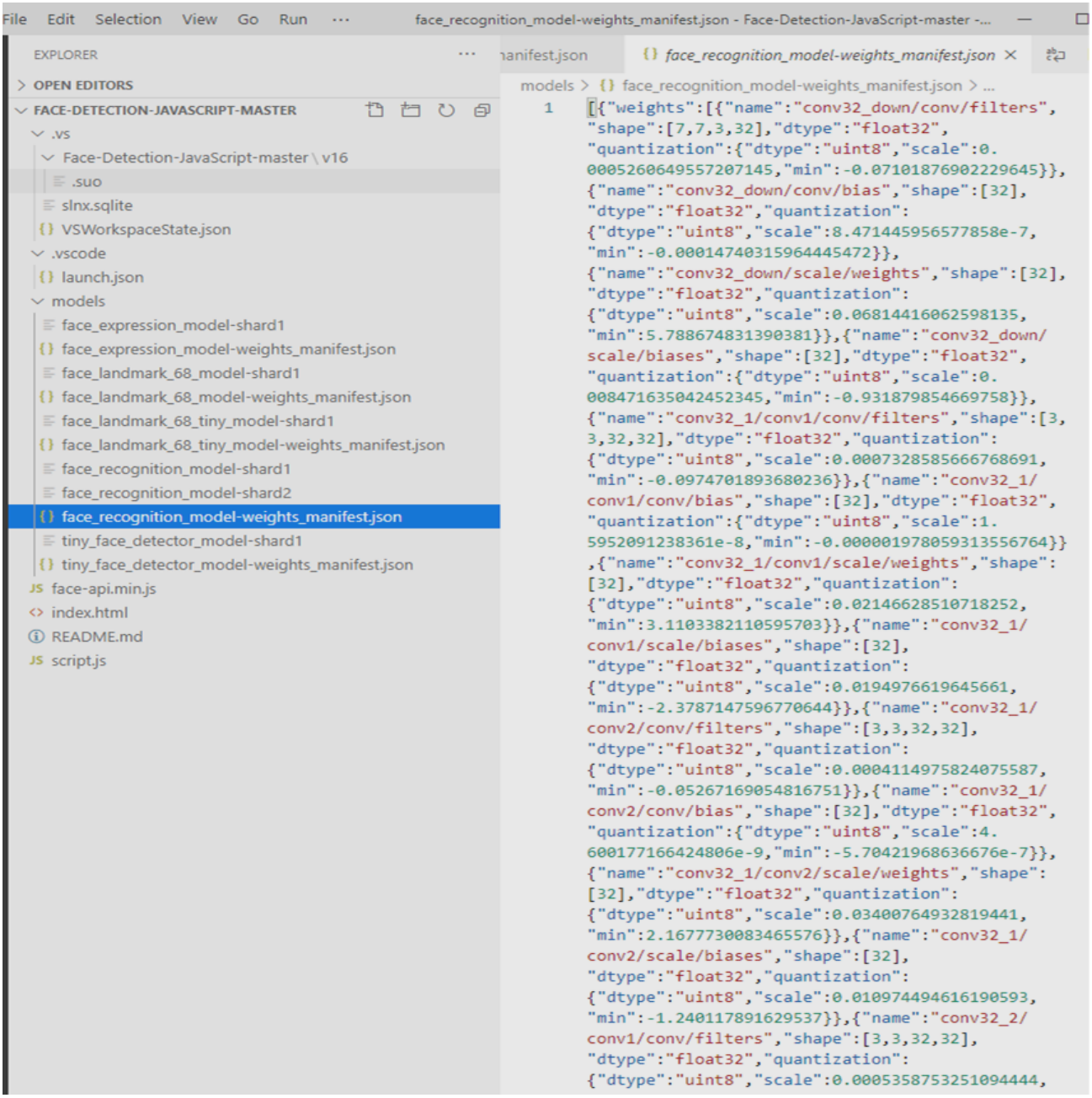


Рисунок 2.6 – Програмний опис реалізації 2D каркасу обличчя

Завантаження зображень у систему виконується за допомогою методу const imageUpload = document.getElementById('imageUpload').

Після завантаження зображення, виконується порівняння моделей зображення, створюється пошук тестових (кінечних) моделей з їх початковим шаблонами. Це все описується у блоці async function start().

Повний цикл навчання нейронної мережі описується в графічному представлені.

Завантаження зображень у систему виконується за допомогою const imageUpload = document.getElementById('imageUpload').

Після завантаження зображення, виконується порівняння моделей зображення, створюється пошук тестових (кінечних) моделей з їх початковим шаблонами. Це все описується у блоці async function start().

Папка test_images зберігає у собі тестові зображення, за допомогою яких наприкінці буде перевірка навчання системи.

Створення системи ідентифікації. Для того аби позначити обличчя на зображенні та ідентифікувати його, необхідно прописати метод Canvas (рис. 2.7).

```
        canvas {
        position: absolute; top: 0;
        left: 0;
}
```

Рисунок 2.7

Бібліотека Face-api.min.js. Цей репозиторій зберігає у собі усі необхідні дані, за допомогою яких відбувається ідентифікація особи на зображенні.

Для порівняння осіб можна використовувати точковий 2D каркас одного і того ж положення особи.

Всі ознаки (відстані) повинні бути безрозмірні (нормалізованості), тобто, співвіднесені до якогось розміру (відстані). Припускаю, що найбільш

підходящий для цього розмір – відстань між центрами кутових точок очей.

Справа в тому, що кутові точки очей розсуваються (зближуються) при реагуванні на зміну кольору, вираженні подиву, моргання і т.п. Відстань між центрами очей нівелює ці коливання і тому більш переважно.

За основу необхідно взяти відстань від верхньої точки перенісся до нижньої точки підборіддя.

Перш ніж формувати ознаки для навчання і порівняння, необхідно відфільтрувати отримані відеозахоплення точкові каркаси осіб, які з суб'єктивних чи об'єктивних причин не є правильне фронтальне зображення особи (анфас).

Пряма, яка проходить через крайні точки очей (лінія очей), перпендикулярна прямий, яка проходить через крайні точки носа (лінія носа).

Лінія очей паралельна прямій, яка проходить через точки куточків рота (лінія рота).

Дотримується симетрія зазначених вище точок щодо лінії носа. Кутові точки очей (зовнішні і внутрішні) знаходяться на одній прямій.

2.2 Опис архітектури нейронної мережі

Ідентифікація об'єкта, а саме обличчя, створюється у face-api.js. Це процес є асинхронним, та виконується у методі Promise (рис. 2.8).

```
Promise.all([ faceapi.nets.tinyFaceDetector.loadFromUri('/models'),
faceapi.nets.faceLandmark68Net.loadFromUri('/models'),
faceapi.nets.faceRecognitionNet.loadFromUri('/models'),
faceapi.nets.faceExpressionNet.loadFromUri('/models')
]).then(startVideo)
```

Рисунок 2.8

Слід розглянути детальніше моделі розглянуті у методі Promise.

Модель tinyFaceDeitector ініціалізується перша та виконує аналіз об’єктів у просторі камери faceapi.nets.tinyFaceDetector.loadFromUri('/models').

Модель faceLandmark68Net (див. рис. 2.9) реалізує оцінку відстані між ключовими точками обличчя за методом із 68 точок.

Оцінки створюється за допомогою коефіцієнтів weights faceapi.nets.faceLandmark68Net.loadFromUri('/models').

Метод faceRecognitionNet ініціалізується у режиме потоку відео (онлайн у момент включення послідовно двох попередніх моделей), та виконує аналіз положення обличчя у просторі та відповідно цього разом із обличчям переміщує шаблон із точок моделі faceLandmark68Net faceapi.

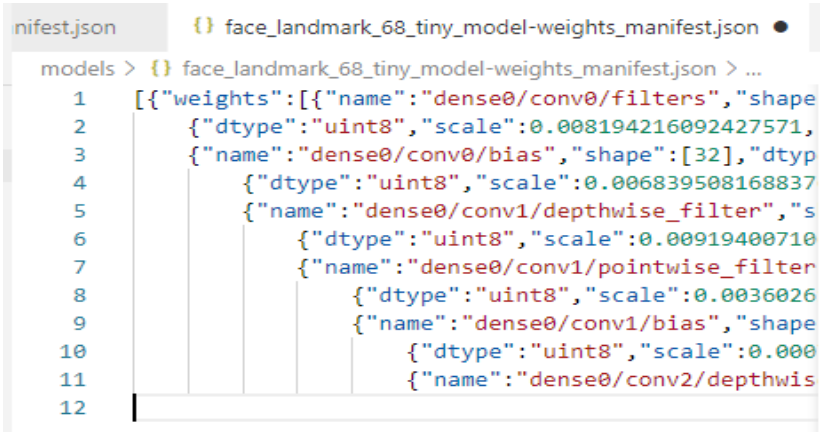


Рисунок 2.9 – Відображення валів моделі faceLandmark68Net

Метод faceExpressionNet виконує обчислення положення губ та виразу обличчя (поведінки та настрою), це виконується за допомогою вагів методу faceapi.nets.faceExpressionNet.loadFromUri('/models') (рис. 2.10).

Останнім етапом є відображення рамки розпізнавання обличчя у виді синього квадрату, у якому є такі показники як, процент якості відображення (0,1 – 0,99) показник точності ідентифікації (0,1 – 0,99) необхідних параметрів, наприклад, настрій (neutral, happy, angry, adjusted).

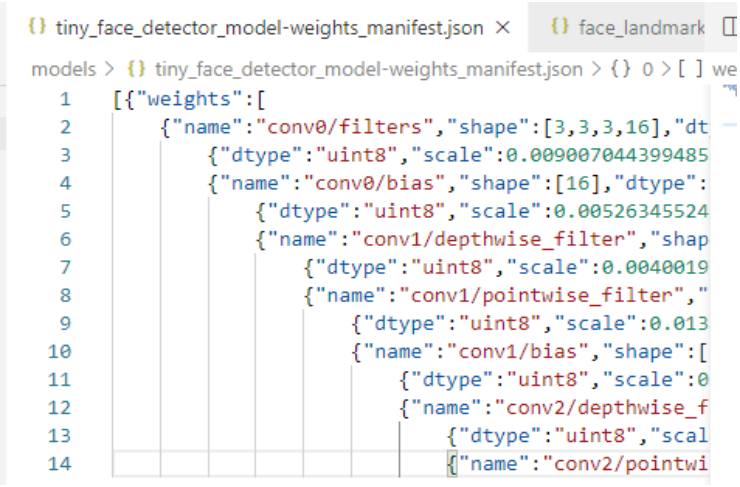


Рисунок 2.10 – Відображення вагів моделі faceLandmark68Net

Розроблена система дозволяє розпізнавати обличчя з маскою, проте точність та якість все ж залишаються питанням, яке у подальшому буде вирішено з модернізацією системи, та впровадженням до системи нових необхідних параметрів.

2.3 Проєктування бази даних та інтерфейсу системи

База даних буде складатися з ключових точок розпізнавання облич.

Ключовими точками можуть бути куточки очей, губ, кінчик носа, центр ока тощо (див. рис. 2.11). Як ключові області можуть бути прямокутні області, що включають у себе: очі, ніс, рот.

У процесі розпізнавання порівнюються ознаки невідомої особи з ознаками, що зберігаються у базі.

Задача знаходження ключових точок наближається за трудомісткістю безпосередньо до розпізнавання, і правильне знаходження ключових точок на зображенні багато в чому визначає успіх розпізнавання.

Тому зображення обличчя людини має бути без завад, що заважають

процесу пошуку ключових точок. До таких завад відносять окуляри, бороди, прикраси, елементи зачіски й макіяжу. Освітлення бажано рівномірне й однакове для всіх зображень.

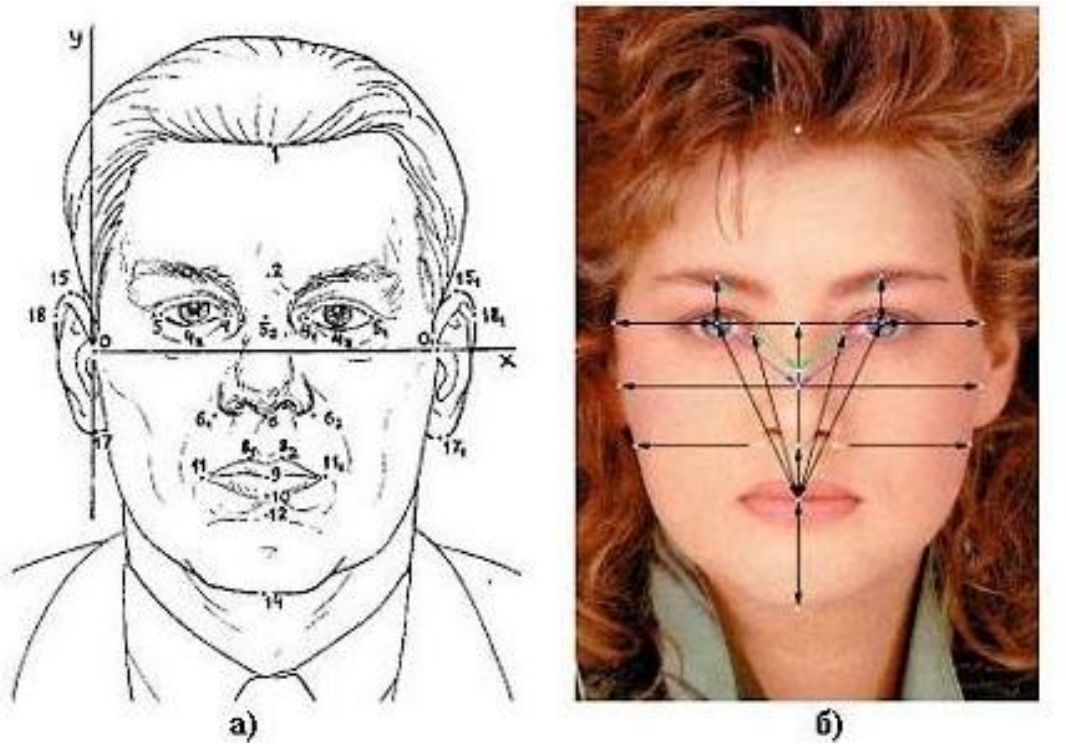


Рисунок 2.11 – Ідентифікаційні точки та відстані: а) використовувані при криміналістичній фотоекспертизі; б) найбільш часто використовувані при побудові автоматизованих систем ідентифікації

Крім того, зображення обличчя повинно мати фронтальний ракурс, можливо з невеликими відхиленнями. Вираз обличчя має бути нейтральним. Це пов’язано з тим, що у більшості методів немає моделі врахування таких змін.

Таким чином, даний метод висуває суворі вимоги до умов зйомки, потребує надійного механізму знаходження ключових точок для загального випадку. Крім того, потрібне застосування більш досконалих методів класифікації або побудови моделі змін.

У загальному випадку цей метод не найоптимальніший, проте для деяких

специфічних завдань перспективний. До таких завдань можна віднести документний контроль, коли потрібно порівняти зображення обличчя, отримане у поточний момент з фотографією у документі. При цьому інших зображень цієї людини немає, і отже механізми класифікації, засновані на аналізі тренувального набору, недоступні.

Порівняння еталонів (*Template Matching*) полягає у виділенні областей особи на зображенні (рис. 2.12), й у наступному порівнянні цих областей для двох різних зображень. Кожна область, що збігається, збільшує міру подібності зображень. Це також один із історично перших методів розпізнавання людини за зображенням особи. Для порівняння областей використовуються найпростіші алгоритми як попіксельне порівняння.

Рисунок 2.12 – Порівнювальні області – еталони обличчя

Недолік цього методу полягає у тому, що він вимагає багато ресурсів як для зберігання ділянок, так і для їх порівняння. З причини того, що використовується найпростіший алгоритм порівняння, зображення повинні бути зняті у суворо встановлених умовах: не допускається помітних змін ракурсу, освітлення, емоційного вираження та ін.

Процес ідентифікації особи за параметрами обличчя умовно можна поділити на дві фази – пошук області обличчя на зображенні та отримання ознак обличчя з цієї області.

2.4 Висновки до розділу 2

У загальному випадку, ідентифікація за обличчям складається з декількох етапів:

– отримання зображення та перетворення його у напівтонове зображення;
– пошук області обличчя на зображення;
– отримання ознак обличчя (запис їх до бази);
– отримання ознак обличчя та їх порівняння з еталонними ознаками.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ІДЕНТИФІКАЦІЇ ОБЛИЧЧЯ ЛЮДИНИ

3.1 Реалізація функціоналу системи

Щоб знайти обличчя на зображенні, потрібно зробити його чорно-білим, оскільки кольорові дані для даного процесу не потрібні (рис. 3.1).

Рисунок 3.1 – Переведення у чорно-білий

Цей процес виконується за допомогою функції переведення зображення у чорно-білий колір, що наявна в бібліотеці OpenCV (рис. 3.2).

```
def findEncodings(images):
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Рисунок 3.2 – Функції для переведення у чорно-білий

Потім розглядається кожен піксель на зображенні по одному. У даному випадку для кожного окремого пікселя важливо розглянути пікселі, які безпосередньо його оточують (рис. 3.3).

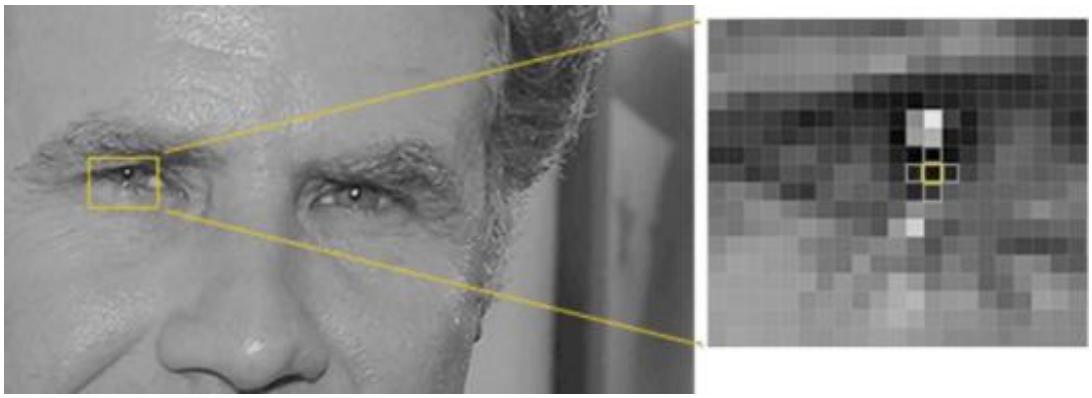


Рисунок 3.3 – Аналіз пікселів

Мета цього процесу – з’ясувати, наскільки темним є поточний піксель порівняно з пікселями, що безпосередньо його оточують. Далі розміщується стрілка, показуючи, в якому напрямку зображення стає темнішим (рис. 3.4).

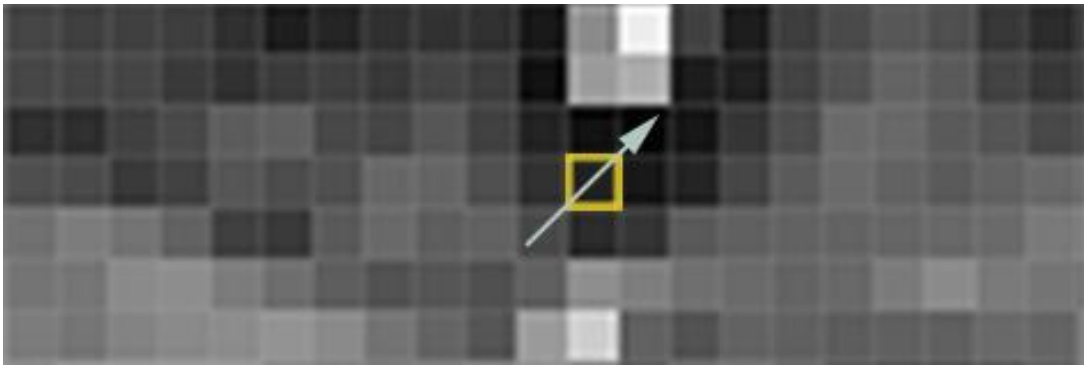


Рисунок 3.4 – Градієнти пікселів

Повторюючи цей процес для кожного пікселя на зображенні, у результаті кожен піксель буде замінено стрілкою. Ці стрілки називаються градієнтами, і вони показують потік від світлого до темного пікселя по всьому зображенню (рис. 3.5–3.6).

Рисунок 3.5 – Градієнти пікселів зображення

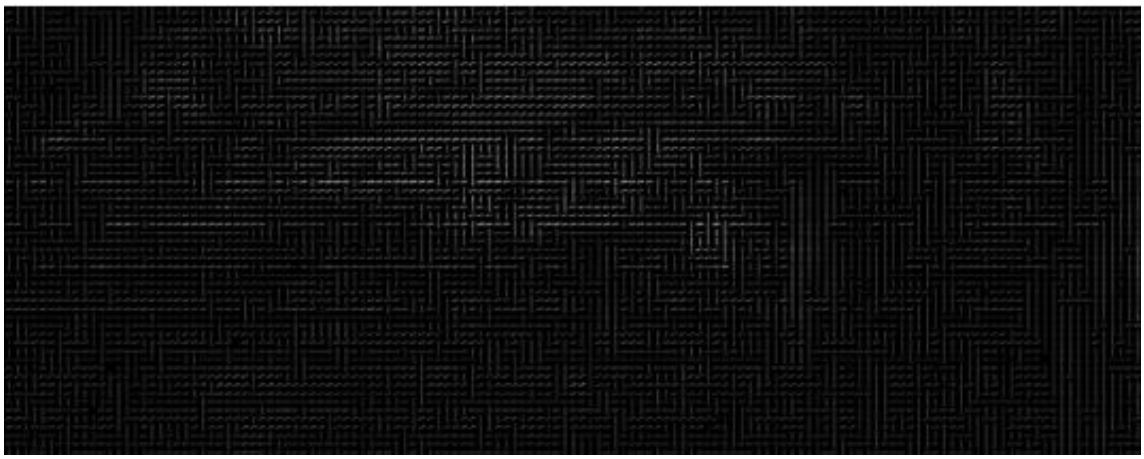


Рисунок 3.6 – Градієнти пікселів зображення

Є вагома причина для заміни пікселів градієнтами. Якщо проаналізувати безпосередньо пікселі, то дуже темні зображення і дуже світлі зображення однієї і тієї ж людини матимуть абсолютно різні значення пікселів. Але враховуючи напрямок зміни яскравості, як темні, так і яскраві зображення в кінцевому підсумку отримають однакове точне зображення.

Але збереження градієнта для кожного пікселя дає занадто багато деталей що у підсумку створює проблеми для подальшого розпізнавання.

Тому зображення розбивається на маленькі квадратики розміром 16x16 пікселів кожен. У кожному квадраті обраховується, скільки градієнтів вказують

загальний напрямок (скільки вказують вгору, вгору-вправо, вправо тощо). Тоді цей квадрат на зображенні замінюється напрямками стрілок, що переважали.

Кінцевим результатом є зображення, перетворене з оригінального на дуже просте, яке фіксує основну структуру обличчя (рис. 3.7).

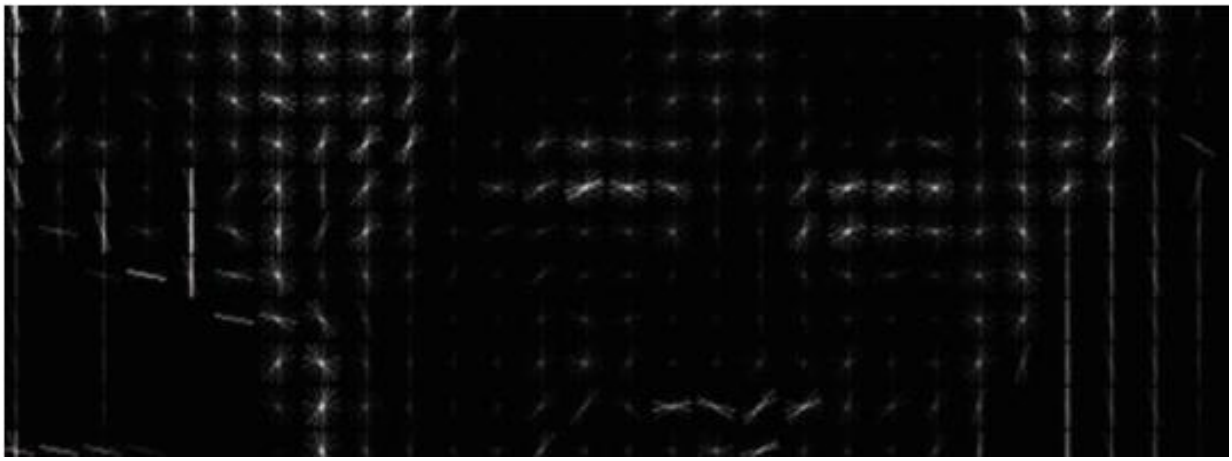


Рисунок 3.7 – Структура обличчя

Щоб знайти обличчя на цьому зображенні HOG, все, що потрібно зробити, це знайти частину іншого зображення, яка виглядає найбільш схожою на тепер відомий шаблон HOG, витягнутий з купи інших навчальних облич (рис. 3.8).

Рисунок 3.8 – Порівняння ознак

Для виконання попередніх пунктів застосовується функція знаходження обличчя у бібліотеці Face recognition, що використовує описаний метод детектування (рис. 3.9).

```
faceLoc = face_recognition.face_locations(imgRus)[0]
encodeElon = face_recognition.face_encodings(imgRus)[0]
cv2.rectangle(imgRus, (faceLoc[3], faceLoc[0]),
              (faceLoc[1], faceLoc[2]), (255, 0, 255), 2)

faceLocTest = face_recognition.face_locations(imgTest)[0]
encodeTest = face_recognition.face_encodings(imgTest)[0]
cv2.rectangle(imgTest,(faceLocTest[3],faceLocTest[0]),
              (faceLocTest[1],faceLocTest[2]),(255,0,255),2)
```

Рисунок 3.9 – Знаходження обличчя

Технологія дозволяє легко знаходити обличчя на будь-якому зображенні. Також окрім OpenCV використовувались інші бібліотеки:

– CV – це модуль обробки зображень та комп’ютерного зору;
– HighGUI – модуль для введення/виведення зображень і відео, створення інтерфейсу користувача;
– ML – вбудовані алгоритми машинного навчання.

Це короткий опис деяких.

3.2 Процес побудови та тренування моделі

Дослідники виявили, що найточніший підхід – це дозволити комп’ютеру самостійно з’ясувати які вимірювання збирати. Алгоритм глибокого навчання краще, ніж люди, з’ясовує, які частини обличчя важливо виміряти.

Рішення полягає у підготовці глибокої згорткової нейронної мережі. Але замість того, щоб навчити мережу розпізнавати об’єкти зображень, як це було

минулого разу, потрібно навчити її генерувати 128 значень для кожного обличчя. Тренувальний процес відбувається під час перегляду одночасно 3 зображень облич:

– тренувальне зображення обличчя відомої людини;
– ще одне зображення тієї ж відомої людини;
– зображення зовсім іншої людини.

Потім алгоритм розглядає виміряні значення, які він генерує в даний час для кожного з цих трьох зображень.

Потім він дещо налаштовує нейронну мережу, щоб переконатися, що вимірювання, які він генерує для №1 та №2, дещо ближчі, ніж вимірювання для №2 та №3, що більше відрізняються (рис. 3.10).

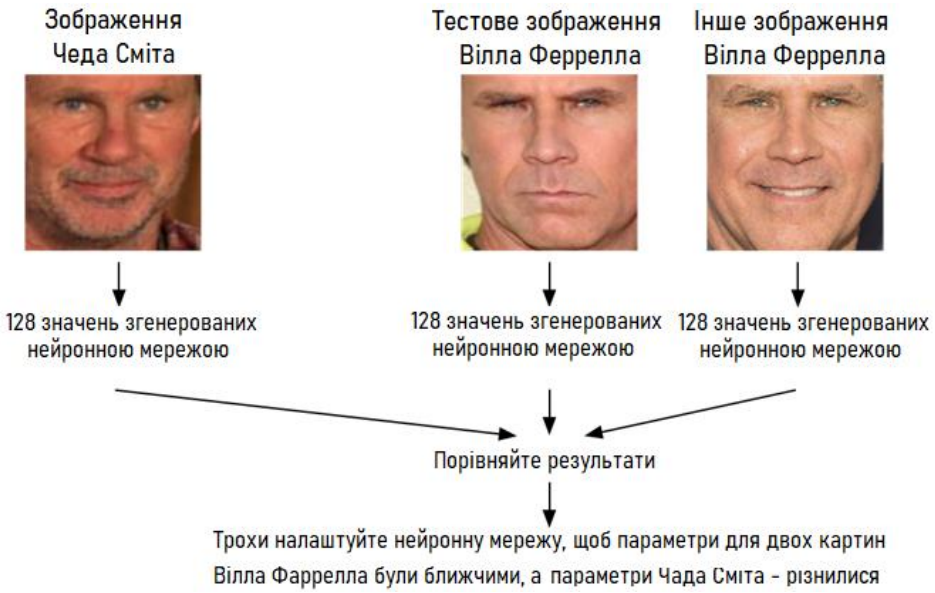


Рисунок 3.10 – Тренування нейронної мережі

Повторивши цей крок мільйони разів для мільйонів зображень тисяч різних людей, нейронна мережа вчиться надійно генерувати 128 параметрів для кожної людини.

Будь-які десять різних зображень однієї і тієї ж людини повинні давати приблизно однакові виміри.

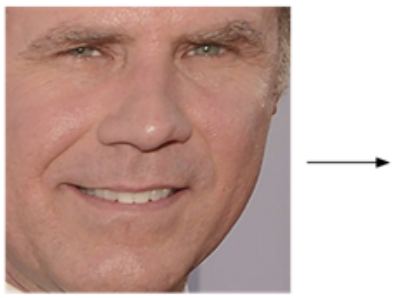
Люди, які вивчають машинне навчання, називають 128 параметрів кожного обличчя «вбудовуванням». Ідея скорочення складних необроблених даних, таких як зображення, у перелік комп'ютерних чисел, дуже часто виникає в машинному навчанні.

Точний підхід до обличчя, що використаний у цій роботі, був винайдений у 2015 році дослідниками з Google, але існує багато подібних підходів.

Кодування нашого зображення обличчя – це процес підготовки згорткової нейронної мережі для виведення вбудованих граней, що вимагає багато даних та потужності комп'ютера. Навіть із дорогою відеокартою NVidia Telsa, щоб отримати хорошу точність, потрібно близько 24 годин безперервного тренування [10].

Але після того, як мережа пройшла навчання, вона може генерувати виміри для будь-якого обличчя, навіть такого, якого вона ще ніколи не бачила. Тож цей крок потрібно зробити лише один раз.

Співробітники з OpenFace пройшли процес тренування, і вони опублікували кілька навчених мереж, одна з яких використана у даній роботі. Тепер потрібно провести свої зображення обличчя через попередньо навчену мережу, щоб отримати 128 вимірювань для кожного обличчя (рис. 3.11).



0.097496084868908	0.045223236083984	-0.1281466782093	0.032084941864014
0.12529824674129	0.060309179127216	0.17521631717682	0.020976085215807
0.030809439718723	-0.01981477253139	0.10801389068365	-0.00052163278451189
0.036050599068403	0.065554238855839	0.0731306001544	-0.1318951100111
-0.097486883401871	0.1226262897253	-0.029626874253154	-0.0059557510539889
-0.0066401711665094	0.036750309169292	-0.15958009660244	0.043374512344599
-0.14131525158882	0.14114324748516	-0.031351584941149	-0.053343612700701
-0.048540540039539	-0.061901587992907	-0.15042643249035	0.078198105096817
-0.12567175924778	-0.10568545013666	-0.12728653848171	-0.076289616525173
-0.061418771743774	-0.074287034571171	-0.065365232527256	0.12369467318058
0.046741496771574	0.0061761881224811	0.14746543765068	0.056418422609568
-0.12113650143147	-0.21055991947651	0.0041091227903962	0.089727647602558
0.061606746166945	0.11345765739679	0.021352224051952	-0.0085843298584223
0.061989940702915	0.19372203946114	-0.086726233363152	-0.022388197481632
0.10904195904732	0.084853030741215	0.094635948853878	0.020696049556136
-0.019414527341723	0.0064811296761036	0.21180312335491	-0.050584398210049
0.15245945751667	-0.16582328081131	-0.035577941685915	-0.072376452386379
-0.12216668576002	-0.0072777755558491	-0.036901291459799	-0.034365277737379
0.083934605121613	-0.059730969369411	-0.070026844739914	-0.045013956725597
0.087945111095905	0.11478432267904	-0.089621491730213	-0.013955107890069
-0.021407851949334	0.14841195940971	0.078333757817745	-0.17898085713387
-0.018298890441656	0.049525424838066	0.13227833807468	-0.072600327432156
-0.011014151386917	-0.051016297191381	-0.14132921397686	0.0050511928275228
0.0093679334968328	-0.062812767922878	-0.13407498598099	-0.014829395338893
0.058139257133007	0.0048638740554452	-0.039491076022387	-0.043765489012003
-0.024210374802351	-0.11443792283535	0.071997955441475	-0.012062266469002
-0.057223934680223	0.014683869667351	0.05228154733777	0.012774495407939
0.023535015061498	-0.081752359867096	-0.031709920614958	0.069833360612392
-0.0098039731383324	0.037022035568953	0.11009479314089	0.11638788878918
0.020220354199409	0.12788131833076	0.18632389605045	-0.015336792916059
0.0040337680839002	-0.094398014247417	-0.11768248677254	0.10281457751989
0.051597066223621	-0.10034311562777	-0.040977258235216	-0.082041338086128

Рисунок 3.11 – Кодування обличчя

Мережа генерує майже однакові цифри, переглядаючи дві різні картинки однієї людини.

Щоб використати попередньо треновану на потужному обладнанні нейронну мережу, застосовується функція кодування обличчя у бібліотеці Face recognition (рис. 3.12).

```
faceLoc = face_recognition.face_locations(imgRus)[0]
encodeElon = face_recognition.face_encodings(imgRus)[0]
cv2.rectangle(imgRus, (faceLoc[3], faceLoc[0]),
              (faceLoc[1], faceLoc[2]), (255, 0, 255), 2)

faceLocTest = face_recognition.face_locations(imgTest)[0]
encodeTest = face_recognition.face_encodings(imgTest)[0]
cv2.rectangle(imgTest,(faceLocTest[3],faceLocTest[0]),
              (faceLocTest[1],faceLocTest[2]),(255,0,255),2)
```

Рисунок 3.12 – Кодування обличчя

3.3 Тестування програмного комплексу

Останній крок передбачає, що алгоритм знаходить людину в базі даних відомих людей, яка має найближчі виміри до тестового зображення.

Потрібно підготувати класифікатор, який може проводити вимірювання за новим тестовим зображенням і повідомляти, яка відома особа найбільш відповідає. Запуск цього класифікатора займає мілісекунди. Результатом класифікатора є ім’я людини.

Далі використовуються усі зазначені перед цим функції (див. рис. 3.13).

Отже, тепер можна перевірити роботу додатку і визначити, як добре він розпізнає обличчя.

Для цього було взято декілька зображень таких людей як Ілон Маск, Білл Гейтс та Талибов Рустам.

```
while True:
    success, img = cap.read()
    imgS = cv2.resize(img,(0,0),None,0.25,0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)
    facesCurFrame = face_recognition.face_locations(imgS)
    encodesCurFrame = face_recognition.face_encodings(imgS,facesCurFrame)
    for encodeFace,faceLoc in zip(encodesCurFrame,facesCurFrame):
        matches = face_recognition.compare_faces(encodeListKnown,encodeFace)
        faceDis = face_recognition.face_distance(encodeListKnown,encodeFace)
        matchIndex = np.argmin(faceDis)
        if faceDis[matchIndex] < 0.50:
            name = classNames[matchIndex].upper()
            markAttendance(name)
        else:
            name = 'Unknown'
        y1, x2, y2, x1 = faceLoc
        y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
        cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)
        cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
    cv2.imshow('Webcam',img)
    cv2.waitKey(1)
```

Рисунок 3.13 – Розпізнавання обличчя

Перше контрольне фото виглядає так (рис. 3.14).

Рисунок 3.14 – Обличчя для розпізнавання

Далі перший набір тестових фото, що були розпізнані (рис. 3.15 – 3.16).

Рисунок 3.15 – Розпізнане обличчя

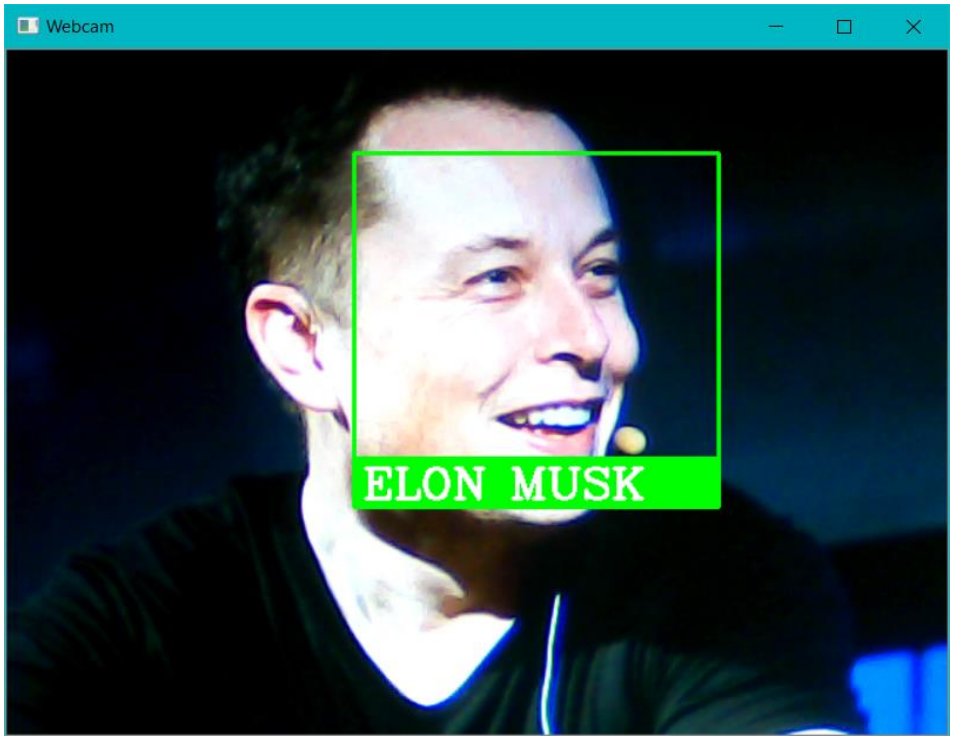


Рисунок 3.16 – Розпізнане обличчя

Друге контрольне фото виглядає так (рис. 3.17).

Рисунок 3.17 – Обличчя для розпізнавання

Для проведення тестування, складаються тест-плани. Перелік умов, при яких буде відбуватися тестуватися:

– визначення особи занесеної до бази;
– визначення особи не занесеної до бази;
– визначення особи при різному освітленні;
– визначення особи при різному куті розміщення обличчя;
– визначення особи при різних виразах обличчя;
– визначення особи при наявності бороди, вусів, окулярів, маски.

Для перевірки складаються тест-кейси, за якими і перевіряється програма (табл. 3.1 – 3.13).

Таблиця 3.1 – Тест-кейси 1 для перевірки програми

Опис		Визначення особи занесеної до бази
Передумови		Завантаження фото
№	Дія	Очікуваний результат
1	Дочекатися результатів розпізнавання	Особа розпізнана (виведено ім’я)

Таблиця 3.2 – Тест-кейси 2 для перевірки програми

Опис		Визначення особи не занесеної до бази
Передумови		Завантаження фото
№	Дія	Очікуваний результат
1	Дочекатися результатів розпізнавання	Особа не розпізнана (виведено «unknown»)

Таблиця 3.3 – Тест-кейси 3 для перевірки програми

Опис		Визначення особи при світлі (лампа денного світла 900Lm)
Передумови		Завантаження фото
№	Дія	Очікуваний результат
1	Дочекатися результатів розпізнавання	Особа розпізнана

Таблиця 3.4 – Тест-кейси 4 для перевірки програми

Опис		Визначення особи в темряві
Передумови		Завантаження фото
№	Дія	Очікуваний результат
1	Дочекатися результатів розпізнавання	Особа не розпізнана

Таблиця 3.5 – Тест-кейси 5 для перевірки програми

Опис		Визначення особи при куті розміщення обличчя 16 – 26%
Передумови		Завантаження фото
№	Дія	Очікуваний результат
1	Дочекатися результатів розпізнавання	Особа розпізнана

Таблиця 3.6 – Тест-кейси 6 для перевірки програми

Опис		Визначення особи куті розміщення обличчя 26 – 40%
Передумови		Завантаження фото
№	Дія	Очікуваний результат
1	Дочекатися результатів розпізнавання	Особа частково розпізнана

Таблиця 3.7 – Тест-кейси 7 для перевірки програми

Опис		Визначення особи при наявності бороди
Передумови		Завантаження фото
№	Дія	Очікуваний результат
1	Дочекатися результатів розпізнавання	Особа розпізнана

Таблиця 3.8 – Тест-кейси 8 для перевірки програми

Опис		Визначення особи при наявності вусів
Передумови		Завантаження фото
№	Дія	Очікуваний результат
1	Дочекатися результатів розпізнавання	Особа розпізнана

Таблиця 3.9 – Тест-кейси 9 для перевірки програми

Опис		Визначення особи при наявності окулярів з прозорими лінзами
Передумови		Завантаження фото
№	Дія	Очікуваний результат
1	Дочекатися результатів розпізнавання	Особа розпізнана

Таблиця 3.10 – Тест-кейси 10 для перевірки програми

Опис		Визначення особи при наявності окулярів з темними лінзами
Передумови		Завантаження фото
№	Дія	Очікуваний результат
1	Дочекатися результатів розпізнавання	Особа не розпізнана

Таблиця 3.11 – Тест-кейси 11 для перевірки програми

Опис		Визначення особи при наявності медичної маски що покриває ніс
Передумови		Завантаження фото
№	Дія	Очікуваний результат
1	Дочекатися результатів розпізнавання	Особа не розпізнана

Таблиця 3.12 – Тест-кейси 12 для перевірки програми

Опис		Визначення особи при наявності медичної маски що не покриває ніс
Передумови		Завантаження фото
№	Дія	Очікуваний результат
1	Дочекатися результатів розпізнавання	Особа розпізнана

Таблиця 3.13 – Тест-кейси 13 для перевірки програми

Опис		Визначення особи при наявності усмішки
Передумови		Завантаження фото
№	Дія	Очікуваний результат
1	Дочекатися результатів розпізнавання	Особа розпізнана

Тестування показало, що програма добре розпізнає обличчя у базі і виводить імена осіб, а також виводить напис «Unknown», якщо особа відсутня.

При наявності освітлення(~900Lm), програма розпізнає обличчя. При поганому освітленні або за його відсутності обличчя майже не розпізнаються.

При горизонтальному куті розміщення обличчя 16 – 26% програма розпізнає обличчя. При горизонтальному куті розміщення обличчя 26 – 40% програма може не розпізнати обличчя.

При наявності бороди, вусів, окулярів з прозорими лінзами програма розпізнає обличчя. При наявності окулярів з темними лінзами обличчя не розпізнаються. При наявності медичної маски обличчя не розпізнається, але може бути розпізнане, якщо маска не покриває ніс.

При наявності усмішки програма розпізнає обличчя.

3.4 Висновки до розділу 3

В розділі 3 описано логіка розробки та функціонування програмного додатку нейромережевого розпізнавання облич. Також проведено тестування роботи розробленого застосунку. При дотриманні всіх умов для розпізнавання обличчя, які були виявленні в пункту 3.3, вірогідність розпізнання обличчя варіюється між 70–90%.

ВИСНОВКИ

Здатність до навчання є невід’ємною частиною властивості нейронних мереж. Навчання посилює працездатність системи у відповідності до задач та правил, які покладені у основу мережі. Нейромережа навчається за принципом інтерактивного процесу корегування синаптичних ваг нейронів. Робота мережі має такий вид: вона повинна отримувати знання із навколишнього середовища під час кожної ітерації процесу свого навчання.

У магістерській роботі розглянуто особливості актуальних методів ідентифікації, а також принципів навчання нейромереж, докладно описано головні частини згорткових нейромереж.

Проаналізовано актуальні архітектури нейромережевих систем, методи навчання та засоби покращення характеристик систем.

Розроблена програмна реалізація нейронної мережі ідентифікації на основі технології нейромережевої системи реального часу. Для цього використано Face–API.js та бібліотеки CV, HighGUI, ML, та інші, що дозволяє проєктувати архітектури мереж та здійснювати їх налаштування.

Результатом розробки є нейромережева система реального часу.

Слід зазначити, що система може бути модифікована у разі появи нових завдань ідентифікації.

Також є можливість завантажувати свої обличчя для розпізнавання, після запуску програми, точність розпізнавання може мінятись в залежності від чіткості, затемненості та куту обличчя, При наявності маски обличчя не розпізнається, але може бути розпізнане, якщо маска не покриває ніс та очі.

ПЕРЕЛІК ПОСИЛАНЬ

1. ДСТУ 3008-2015. Звіти у сфері науки і техніки. Структура та правила оформлювання. [На заміну ДСТУ 3008-95. Чинний від 2017-07-01]. Київ : Держстандарт України, 2016. 31 с. (Інформація та документація).

2. Невлюдов І. Ш. Дипломне проектування для студентів усіх форм навчання спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології» : навч. посіб. / І. Ш. Невлюдов, А. О. Андрусевич, О. В. Токарєва, Г. В. Пономарьова. Київ – 58, пр. Космонавта Комарова, 1, 2016. 320 с.

3. Методичні вказівки «З розробки й оформлення магістерської атестаційної роботи для студентів другого (магістерського) рівня вищої освіти галузі знань 15 Автоматизація та приладобудування за спеціальністю 151 Автоматизація та комп'ютерно-інтегровані технології освітні програми URL: https://acit.kname.edu.ua/images/%D0%BE%D0%BA_9_14.10.2020_compressed.pdf (дата звернення: 10.10.2023).

4. «Автоматизоване управління технологічними процесами», «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / упоряд. І. Ш. Невлюдов, В. В. Косенко, В. В. Євсєєв. Харків : ХНУРЕ, 2019. 55 с.

5. Положення про протидію академічному плагіату в ХНУРЕ. URL: https://nure.ua/wp-content/uploads/Main_Docs_NURE.html (дата звернення: 29.09.2023).

6. Невлюдов І. Ш. Основи наукових досліджень : навч. посібник / І. Ш. Невлюдов, Ю. М. Олександров, А. О. Андрусевич, О. О. Чала. Кривій Ріг : Криворізький коледж НАУ, 2019. 396 с.

7. Невлюдов І. Ш. Техніко-економічне обґрунтування інженерних рішень автоматизованому виробництві. Кривий Ріг : Криворізький коледж НАУ,

2019. 448 с.

8. Izmerov M. Neurocomputer Modeling Of Contact Rigidity. *Bulletin of Bryansk state technical university*. 2016. 10 с.

9. Srivastava N., Hinton G. E. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*. 2014. Vol. 15. № 1. P. 1929–1958.

10. Бодянський Є. В., Кучеренко Є. І. Нейро-фаззі моделі в системах штучного інтелекту. Харків : ХНУРЕ, 2016. 177 с.

11. Невлюдов І. Ш., Цимбал О. М., Мілютіна С. С. Розпізнавання голосових команд за допомогою багатошарового перцептрона. *Східно-європейський журнал передових технологій*. 2006. № 3/2 (16). С. 13–16.

12. Bergstra J., Bengio Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*. 2022. Vol. 13. № 1. P. 281–305.

13. Cootes T. Active appearance models / T. Cootes, G. J. Edwards, C. J. Taylor, H. Burkhardt, B. Neumann. *5 th European Conference on Computer Vision, Springer*. 2018. Vol. 2. P. 484–498.

14. Shastry A. P. Convolutional Neural Networks. URL: https://medium.com/@aparnack/convolutional-neural-networks-part-1-2aeb17fc208c (дата звернення: 29.09.2023).

15. Паскарюк Д. О. Розпізнавання образів за допомогою нейроних мереж. *Збірник студентських наукових статей* / редкол. : І. Ш. Невлюдов та ін. Харківський національний університет радіоелектроніки, 2020. Вип. 2. С. 74–77.

16. Семенець В. В., Синотин А. М., Колесникова Т. А. Исследование зависимости максимального перегрева радиоэлектронного аппарата от его параметров. *Системи обробки інформації*. 2018. №4. С. 29 –34.

17. Davis B. J. The Recognition of Human Movement UsingTemporal Templates. *IEEE Transactions on Pattern Analysis and MachineIntelligence*. 2021. Vol. 23. No. 3. P. 257–267.

18. Javed O. Tracking and Object Classification for Automated Surveillance. Mubarak Shah Lecture Notes in Computer Science 2002. URL: https://www.researchgate.net/publication/221305117_Tracking_and_Object_Classification_for_Automated_Surveillance (дата звернення: 02.09.2023).

19. Stauffer C., Grimson W. Adaptive Background Mixture Models forReal-Time Tracking. *Proceedings of the IEEE Conference on Computer Visionand Pattern Recognition*. 2019. Vol. 2. P. 246–252.

20. Brown L., Hampapur A., Connell J. IBM Smart Surveillance System (S3): An open and extensible architecture forsmart video surveillance. 2015. URL: https://www.researchgate.net/publication/220465180_IBM_smart_surveillance_system_S3_Event_based_video_surveillance_system_with_an_open_and_extensible_framework (дата звернення: 12.10.2023)

21. Javed O., Shah M. Tracking and Object Classification for AutomatedSurveillance. *Proceedings of the 7th European Conference on ComputerVision*. 2022. Part-IV. P. 343–357.

22. Гайтан О. М., Талибов Р. М. Дослідження можливостей ідентифікації облич методом віоли-джонса в бібліотеці computer vision toolbox системи matlab. URL: http://reposit.nupp.edu.ua/bitstream/PoltNTU/8827/1/Збірник%20тез%20CMSEE-2020-5-6.pdf (дата звернення: 02.09.2023).

ДОДАТОК А

Основний код

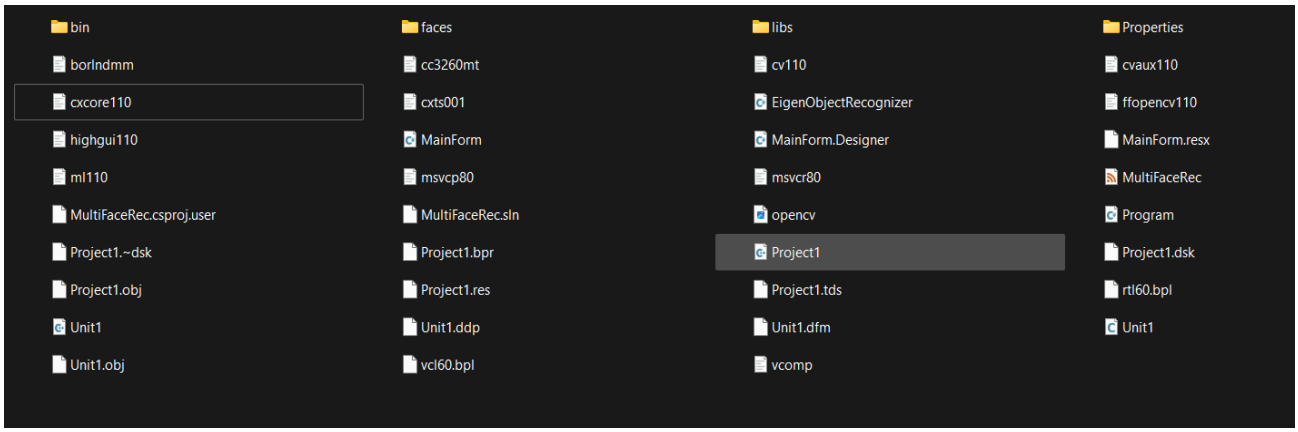


Рисунок А.1

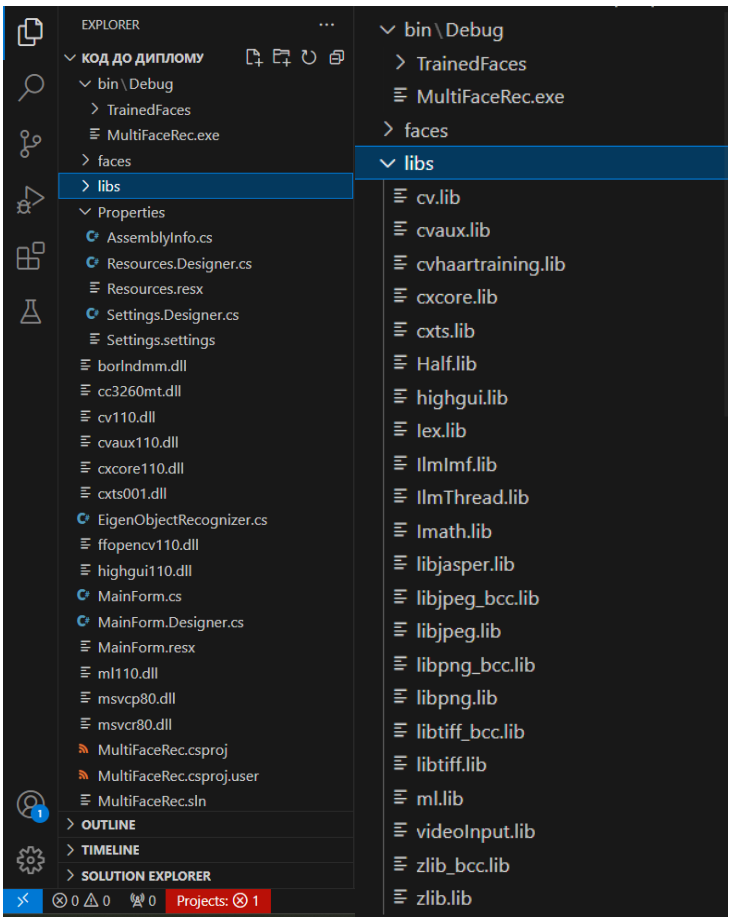


Рисунок А.2

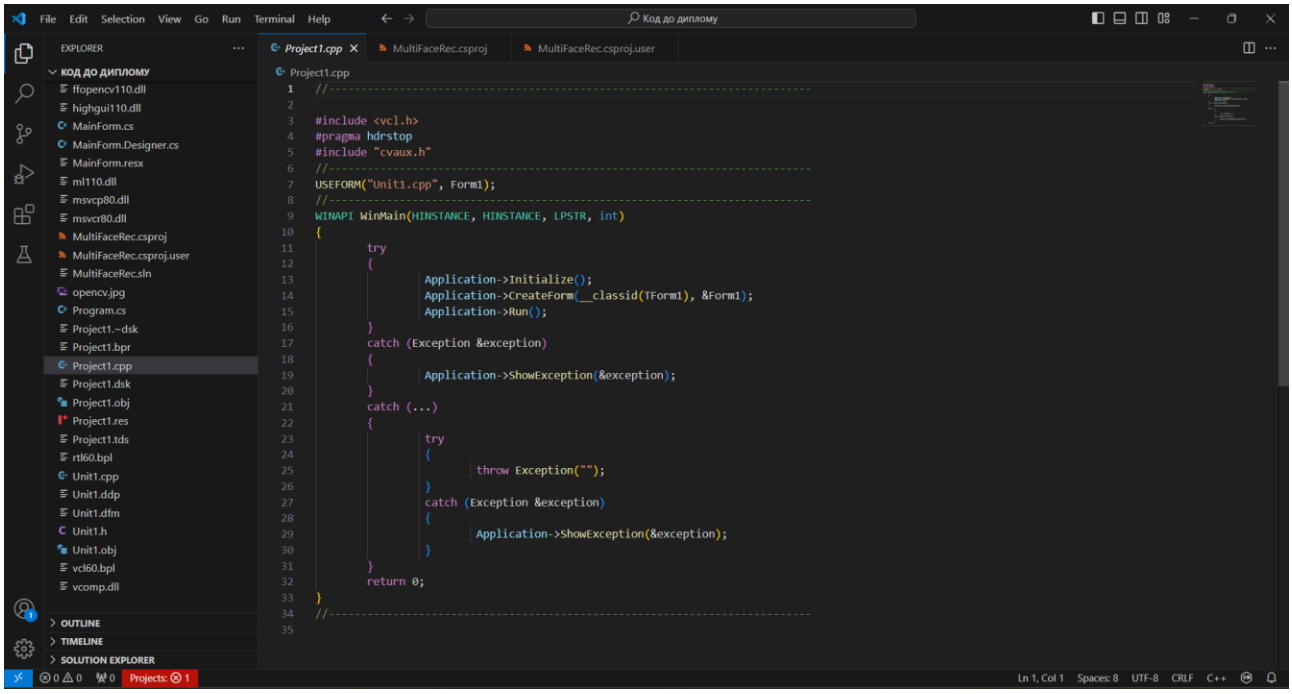


Рисунок А.3

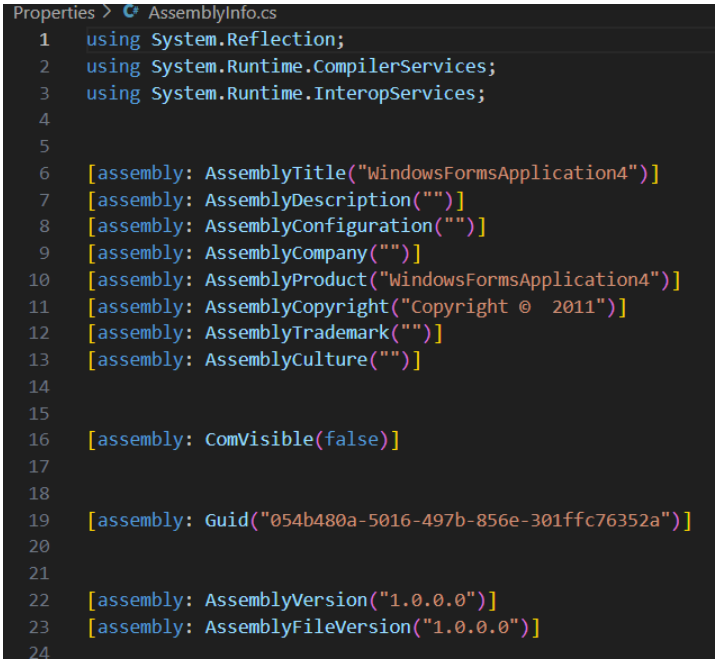


Рисунок А.4

```
//---------------------------------------------------------------------------
                #include <vcl.h>
                #pragma hdrstop
                #include "Unit1.h"
                //---------------------------------------------------------------------------
```

```
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
#define WIDTHBYTES(bits) ((((bits) + 31) / 32) * 4)
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
HBITMAP CreateRGBBitmap(IplImage* _Grab)
        {
char *App;
          LPBITMAPINFO lpbi = new BITMAPINFO;
          lpbi->bmiHeader.biSize = sizeof(BITMAPINFOHEADER);
          lpbi->bmiHeader.biWidth = _Grab->width;
          lpbi->bmiHeader.biHeight =_Grab->height;
          lpbi->bmiHeader.biPlanes = 1;
          lpbi->bmiHeader.biBitCount = 24;
          lpbi->bmiHeader.biCompression = BI_RGB;
             lpbi->bmiHeader.biSizeImage = WIDTHBYTES((DWORD)_Grab->width * 8) * _Grab-
>height;
          lpbi->bmiHeader.biXPelsPerMeter = 0;
          lpbi->bmiHeader.biYPelsPerMeter = 0;
          lpbi->bmiHeader.biClrUsed = 0;
          lpbi->bmiHeader.biClrImportant = 0;
          void* pBits;
          HBITMAP hBitmap = CreateDIBSection(
             NULL,
             lpbi,
             DIB_RGB_COLORS,
             (void **)&pBits,
             NULL,
             0 );
          delete lpbi;
          if ( hBitmap )
App=(char*)pBits;

if(_Grab->nChannels==1)
  {
for (int i=0;i<_Grab->height;i++)
{
 for (int j=0;j<_Grab->width;j++)
 {
  App[_Grab->width*3*(_Grab->height-i-1)+j*3]=_Grab->imageData[_Grab->width*(i)+j];
  App[_Grab->width*3*(_Grab->height-i-1)+j*3+1]=_Grab->imageData[_Grab->width*(i)+j];
  App[_Grab->width*3*(_Grab->height-i-1)+j*3+2]=_Grab->imageData[_Grab->width*(i)+j];
 }
}
  }
if(_Grab->nChannels==3)
  {
for (int i=0;i<_Grab->height;i++)
{
memcpy(App+_Grab->width*3*(_Grab->height-i-1),_Grab->imageData+_Grab->width*3*i,_Grab-
>width*3);
}

  }
return hBitmap;
        }
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
```

```
//
//---------------------------------------------------------------------------
void  APIDrawIpl(int x,int y,IplImage* _Grab,void *HANDLE)
{
HDC hMemDC,hDC;
hDC=GetDC(HANDLE);
hMemDC = CreateCompatibleDC(hDC);
HBITMAP Bitmap=CreateRGBBitmap(_Grab);
SelectObject(hMemDC,Bitmap);
BitBlt(hDC,x,y,_Grab->width,_Grab->height,hMemDC,0,0,SRCCOPY);
DeleteObject(Bitmap);
DeleteDC(hMemDC);
DeleteDC(hDC);
}
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
const unsigned int N_Samples=7;

IplImage* img_load;
IplImage* img_load_ch1[N_Samples];
IplImage* test_img;
IplImage* mean_img=0;
IplImage* result_img=0;
CvSize size;
//****************************************
//
//****************************************
for (int i=0;i<N_Samples;i++)
 {
        img_load = cvLoadImage(("faces//s"+IntToStr(i+1)+"//1.pgm").c_str());
        size=cvSize(img_load->width,img_load->height);
        img_load_ch1[i] = cvCreateImage( size, IPL_DEPTH_8U, 1 );
        cvSplit(img_load,img_load_ch1[i],NULL,NULL,NULL);
        //
        APIDrawIpl(10+(size.width+10)*i,50,img_load,Form1->Handle);
 }
// :)
Form1->Label1->Left=(10+(size.width+10)*N_Samples-Form1->Label1->Width)/2;
Form1->Label1->Visible=true;
Form1->Label2->Top=(50+(size.height+10)-5);
Form1->Label2->Left=(10+(size.width+10)+(size.width+10)*(N_Samples-1)-Form1->Label2-
>Width)/2;
Form1->Label2->Visible=true;
Form1->Label3->Top=(50+(size.height+20)*3-15);
Form1->Label3->Left=(10+(size.width+10)+(size.width+10)*(N_Samples-1)-Form1->Label3-
>Width)/2;
Form1->Label3->Visible=true;
Form1->Label4->Top=(50+(size.height+20)*2-15);
Form1->Label4->Left=(10+(size.width+10)+(size.width+10)*(N_Samples-1)-Form1->Label4-
>Width)/2;
Form1->Label4->Visible=true;
```

```
        //

        //****************************************
        //
        //****************************************
              i=3; //
              //
              img_load = cvLoadImage(("faces//s"+IntToStr(i+1)+"//2.pgm").c_str());
              test_img = cvCreateImage( size, IPL_DEPTH_8U, 1 );
              cvSplit(img_load,test_img,NULL,NULL,NULL);
                APIDrawIpl(10  +  ((size.width+10)*(N_Samples-1))/2,50+(size.height+20)*2,img_load,Form1-
>Handle);
        //****************************************
        //
        //****************************************
        //
        if( !mean_img )
         {
               mean_img = cvCreateImage( size, IPL_DEPTH_32F, 1 );
         }
        //
        if( !result_img )
         {
               result_img = cvCreateImage( size, IPL_DEPTH_8U, 1 );
         }
        //-----------------------------------
        //-----------------------------------
        CvTermCriteria Tc;
        Tc.type=CV_TERMCRIT_NUMBER | CV_TERMCRIT_EPS;

        //
        Tc.max_iter=100;
        Tc.epsilon=0.001;
        //-----------------------------------------------------
        //-----------------------------------------------------
        const int nEigens = N_Samples-1;

        //
        IplImage* eig_img[nEigens];

        CvMat *EigenVals;
        EigenVals = cvCreateMat(1,nEigens,IPL_DEPTH_32F);
        //
        for( i=0;i<N_Samples;i++)
        {
               eig_img[i] = cvCreateImage( size, IPL_DEPTH_32F, 1 );
        }
        //
        cvCalcEigenObjects(N_Samples,                    img_load_ch1                    ,
eig_img,CV_EIGOBJ_NO_CALLBACK,0,0,&Tc,mean_img,EigenVals->data.fl);
        //****************************************
        //
        //****************************************
        double min_val,max_val;
        for (int j=0;j<nEigens;j++)
         {
        cvMinMaxLoc( eig_img[j], &min_val, &max_val);
        if(min_val!=max_val)
        {
```

```
            cvConvertScale( eig_img[j], result_img,255.0/(max_val-min_val),-min_val);
                  APIDrawIpl(10+(size.width+10)*j+(size.width+10)/2,50+(size.height+20),result_img,Form1-
>Handle);
            }
             }
            //-------------------------------------------------------
            //
            //-------------------------------------------------------
            float** coeffs;
            coeffs = new float*[N_Samples];
            for (i=0;i<N_Samples;i++)
            {
            coeffs[i] = new float [nEigens];
            cvEigenDecomposite(                    img_load_ch1[i],                    nEigens,
eig_img,CV_EIGOBJ_NO_CALLBACK,0,mean_img, coeffs[i] );
            }
            //-------------------------------------------------------
            //
            //-------------------------------------------------------
            float projectedTestFace[nEigens];
            cvEigenDecomposite(                    test_img,                    nEigens,
eig_img,CV_EIGOBJ_NO_CALLBACK,0,mean_img,projectedTestFace);
            //-------------------------------------------------------
            double leastDistSq = DBL_MAX;
            int iTrain,iNearest =0;
            for (iTrain=0;iTrain<N_Samples;iTrain++)
            {
            double distSq=0;
             for (i=0;i<nEigens;i++)
             {
             float d_i = projectedTestFace[i] - coeffs[iTrain][i];
             distSq += d_i*d_i/EigenVals->data.fl[i];
             }
            if(distSq<leastDistSq)
            {
            leastDistSq = distSq;
            iNearest = iTrain;
            }
            }
            //****************************************
            //****************************************
            //APIDrawIpl(10+(10+size.width),50+(size.height+20)*2,img_load_ch1[iNearest],Form1->Handle);
            for ( i=0;i<N_Samples;i++)
             {
                  //
            if(i!=iNearest)
            {
            cvLine(img_load_ch1[i],cvPoint(0,0),cvPoint(size.width,size.height),CV_RGB(255,0,0), 3, 8);
            cvLine(img_load_ch1[i],cvPoint(size.width,0),cvPoint(0,size.height),CV_RGB(255,0,0), 3, 8);
            }
                   APIDrawIpl(10+(size.width+10)*i,50+(size.height+20)*3,img_load_ch1[i],Form1->Handle);
             }
            //****************************************
            //****************************************
            for (int k=0;k<N_Samples;k++)
             {
                  cvReleaseImage( &eig_img[k] );
                  cvReleaseImage( &img_load_ch1[k] );
             }
```

```
for (i=0;i<N_Samples;i++)
{
 delete coeffs[i];
}
delete coeffs;
 cvReleaseImage( &img_load);
 cvReleaseImage( &mean_img);
 cvReleaseImage( &test_img);
 cvReleaseImage( &result_img);
}
//-------------------------------------------------------------------------
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;
using System.IO;
namespace MultiFaceRec
{
   public partial class FrmPrincipal : Form
   {

      Image<Bgr, Byte> currentFrame;
      Capture grabber;
      HaarCascade face;
      HaarCascade eye;
      MCvFont font = new MCvFont(FONT.CV_FONT_HERSHEY_TRIPLEX, 0.5d, 0.5d);
      Image<Gray, byte> result, TrainedFace = null;
      Image<Gray, byte> gray = null;
      List<Image<Gray, byte>> trainingImages = new List<Image<Gray, byte>>();
      List<string> labels= new List<string>();
      List<string> NamePersons = new List<string>();
      int ContTrain, NumLabels, t;
      string name, names = null;
      public FrmPrincipal()
      {
         InitializeComponent();
               face = new HaarCascade("haarcascade_frontalface_default.xml");
                try
         {
                                    string Labelsinfo = File.ReadAllText(Application.StartupPath +
"/TrainedFaces/TrainedLabels.txt");
            string[] Labels = Labelsinfo.Split('%');
            NumLabels = Convert.ToInt16(Labels[0]);
            ContTrain = NumLabels;
            string LoadFaces;
            for (int tf = 1; tf < NumLabels+1; tf++)
            {
               LoadFaces = "face" + tf + ".bmp";
                trainingImages.Add(new Image<Gray, byte>(Application.StartupPath + "/TrainedFaces/"
+ LoadFaces));
               labels.Add(Labels[tf]);
            }
         }
         catch(Exception e)
         {
```

```
                MessageBox.Show("Nothing in binary database, please add at least a face", "Triained faces load", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
        private void button1_Click(object sender, EventArgs e)
        {
            grabber = new Capture();
            grabber.QueryFrame();
            Application.Idle += new EventHandler(FrameGrabber);
            button1.Enabled = false;
        }
        private void button2_Click(object sender, System.EventArgs e)
        {
            try
            {
                ContTrain = ContTrain + 1;
                gray = grabber.QueryGrayFrame().Resize(320, 240, Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(
                face,
                1.2,
                10,
                Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
                new Size(20, 20));
                foreach (MCvAvgComp f in facesDetected[0])
                {
                    TrainedFace = currentFrame.Copy(f.rect).Convert<Gray, byte>();
                    break;
                }
                TrainedFace = result.Resize(100, 100, Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                trainingImages.Add(TrainedFace);
                labels.Add(textBox1.Text);
                imageBox1.Image = TrainedFace;
                    File.WriteAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt", trainingImages.ToArray().Length.ToString() + "%");
                for (int i = 1; i < trainingImages.ToArray().Length + 1; i++)
                {
                    trainingImages.ToArray()[i - 1].Save(Application.StartupPath + "/TrainedFaces/face" + i + ".bmp");
                        File.AppendAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt", labels.ToArray()[i - 1] + "%");
                }
                    MessageBox.Show(textBox1.Text + "´s face detected and added :)", "Training OK", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch
            {
                MessageBox.Show("Enable the face detection first", "Training Fail", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
        void FrameGrabber(object sender, EventArgs e)
        {
            label3.Text = "0";

            NamePersons.Add("");
                currentFrame = grabber.QueryFrame().Resize(320, 240, Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                gray = currentFrame.Convert<Gray, Byte>();
```

```
                    MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(
                    face,
                    1.2,
                    10,
                    Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
                    new Size(20, 20));
                    foreach (MCvAvgComp f in facesDetected[0])
                    {
                        t = t + 1;
                                result = currentFrame.Copy(f.rect).Convert<Gray, byte>().Resize(100, 100,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                        currentFrame.Draw(f.rect, new Bgr(Color.Red), 2);
                        if (trainingImages.ToArray().Length != 0)
                        {

                        MCvTermCriteria termCrit = new MCvTermCriteria(ContTrain, 0.001);
                        EigenObjectRecognizer recognizer = new EigenObjectRecognizer(
                          trainingImages.ToArray(),
                          labels.ToArray(),
                          3000,
                          ref termCrit);

                        name = recognizer.Recognize(result);


                                currentFrame.Draw(name, ref font, new Point(f.rect.X - 2, f.rect.Y - 2), new
Bgr(Color.LightGreen));
                        }
                           NamePersons[t-1] = name;
                           NamePersons.Add("");
                        label3.Text = facesDetected[0].Length.ToString();
                    }
                        t = 0;
                    for (int nnn = 0; nnn < facesDetected[0].Length; nnn++)
                    {
                        names = names + NamePersons[nnn] + ", ";
                    }
                    imageBoxFrameGrabber.Image = currentFrame;
                    label4.Text = names;
                    names = "";
                    NamePersons.Clear();
                }
            }
        }
        using System;
        using System.Diagnostics;
        using Emgu.CV.Structure;

        namespace Emgu.CV
        {
          [Serializable]
          public class EigenObjectRecognizer
          {
            private Image<Gray, Single>[] _eigenImages;
            private Image<Gray, Single> _avgImage;
            private Matrix<float>[] _eigenValues;
            private string[] _labels;
            private double _eigenDistanceThreshold;
            public Image<Gray, Single>[] EigenImages
```

```
        {
          get { return _eigenImages; }
          set { _eigenImages = value; }
        }
        public String[] Labels
        {
          get { return _labels; }
          set { _labels = value; }
        }
        public double EigenDistanceThreshold
        {
          get { return _eigenDistanceThreshold; }
          set { _eigenDistanceThreshold = value; }
        }
        public Image<Gray, Single> AverageImage
        {
          get { return _avgImage; }
          set { _avgImage = value; }
        }
        public Matrix<float>[] EigenValues
        {
          get { return _eigenValues; }
          set { _eigenValues = value; }
        }
        private EigenObjectRecognizer()
        {
        }
        public EigenObjectRecognizer(Image<Gray, Byte>[] images, ref MCvTermCriteria termCrit)
          : this(images, GenerateLabels(images.Length), ref termCrit)
        {
        }
        private static String[] GenerateLabels(int size)
        {
          String[] labels = new string[size];
          for (int i = 0; i < size; i++)
            labels[i] = i.ToString();
          return labels;
        }
        public EigenObjectRecognizer(Image<Gray, Byte>[] images, String[] labels, ref MCvTermCriteria
termCrit)
          : this(images, labels, 0, ref termCrit)
        {
        }
            public EigenObjectRecognizer(Image<Gray, Byte>[] images, String[] labels, double
eigenDistanceThreshold, ref MCvTermCriteria termCrit)
        {
          Debug.Assert(images.Length == labels.Length, "The number of images should equals the number
of labels");
          Debug.Assert(eigenDistanceThreshold >= 0.0, "Eigen-distance threshold should always >= 0.0");
          CalcEigenObjects(images, ref termCrit, out _eigenImages, out _avgImage);
          _eigenValues = Array.ConvertAll<Image<Gray, Byte>, Matrix<float>>(images,
            delegate(Image<Gray, Byte> img)
            {
              return new Matrix<float>(EigenDecomposite(img, _eigenImages, _avgImage));
            });
          _labels = labels;

          _eigenDistanceThreshold = eigenDistanceThreshold;
        }
```

```
        #region static methods
         public static void CalcEigenObjects(Image<Gray, Byte>[] trainingImages, ref MCvTermCriteria termCrit, out Image<Gray, Single>[] eigenImages, out Image<Gray, Single> avg)
        {
          int width = trainingImages[0].Width;
          int height = trainingImages[0].Height;
                IntPtr[] inObjs = Array.ConvertAll<Image<Gray, Byte>, IntPtr>(trainingImages, delegate(Image<Gray, Byte> img) { return img.Ptr; });
          if (termCrit.max_iter <= 0 || termCrit.max_iter > trainingImages.Length)
            termCrit.max_iter = trainingImages.Length;
          int maxEigenObjs = termCrit.max_iter;
          #region initialize eigen images
          eigenImages = new Image<Gray, float>[maxEigenObjs];
          for (int i = 0; i < eigenImages.Length; i++)
            eigenImages[i] = new Image<Gray, float>(width, height);
                IntPtr[] eigObjs = Array.ConvertAll<Image<Gray, Single>, IntPtr>(eigenImages, delegate(Image<Gray, Single> img) { return img.Ptr; });
          #endregion
          avg = new Image<Gray, Single>(width, height);
          CvInvoke.cvCalcEigenObjects(
             inObjs,
             ref termCrit,
             eigObjs,
             null,
             avg.Ptr);
        }
           public static float[] EigenDecomposite(Image<Gray, Byte> src, Image<Gray, Single>[] eigenImages, Image<Gray, Single> avg)
        {
          return CvInvoke.cvEigenDecomposite(
             src.Ptr,
              Array.ConvertAll<Image<Gray, Single>, IntPtr>(eigenImages, delegate(Image<Gray, Single> img) { return img.Ptr; }),
             avg.Ptr);
        }
        #endregion
         public Image<Gray, Byte> EigenProjection(float[] eigenValue)
        {
          Image<Gray, Byte> res = new Image<Gray, byte>(_avgImage.Width, _avgImage.Height);
          CvInvoke.cvEigenProjection(
            Array.ConvertAll<Image<Gray, Single>, IntPtr>(_eigenImages, delegate(Image<Gray, Single> img) { return img.Ptr; }),
             eigenValue,
             _avgImage.Ptr,
             res.Ptr);
          return res;
        }
        public float[] GetEigenDistances(Image<Gray, Byte> image)
        {
          using (Matrix<float> eigenValue = new Matrix<float>(EigenDecomposite(image, _eigenImages, _avgImage)))
            return Array.ConvertAll<Matrix<float>, float>(_eigenValues,
              delegate(Matrix<float> eigenValueI)
              {
                                return (float)CvInvoke.cvNorm(eigenValue.Ptr, eigenValueI.Ptr, Emgu.CV.CvEnum.NORM_TYPE.CV_L2, IntPtr.Zero);
              });
        }
           public void FindMostSimilarObject(Image<Gray, Byte> image, out int index, out float
```

```
eigenDistance, out String label)
                {
                  float[] dist = GetEigenDistances(image);
                  index = 0;
                  eigenDistance = dist[0];
                  for (int i = 1; i < dist.Length; i++)
                  {
                    if (dist[i] < eigenDistance)
                    {
                      index = i;
                      eigenDistance = dist[i];
                    }
                  }
                  label = Labels[index];
                }
                public String Recognize(Image<Gray, Byte> image)
                {
                  int index;
                  float eigenDistance;
                  String label;
                  FindMostSimilarObject(image, out index, out eigenDistance, out label);
                      return (_eigenDistanceThreshold <= 0 || eigenDistance < _eigenDistanceThreshold ) ?
_labels[index] : String.Empty;
                }
              }
            }
            using System.Reflection;
            using System.Runtime.CompilerServices;
            using System.Runtime.InteropServices;
            [assembly: AssemblyTitle("WindowsFormsApplication4")]
            [assembly: AssemblyDescription("")]
            [assembly: AssemblyConfiguration("")]
            [assembly: AssemblyCompany("")]
            [assembly: AssemblyProduct("WindowsFormsApplication4")]
            [assembly: AssemblyCopyright("Copyright ©  2011")]
            [assembly: AssemblyTrademark("")]
            [assembly: AssemblyCulture("")]
            [assembly: ComVisible(false)]
            [assembly: Guid("054b480a-5016-497b-856e-301ffc76352a")]
            [assembly: AssemblyVersion("1.0.0.0")]
            [assembly: AssemblyFileVersion("1.0.0.0")]
            namespace MultiFaceRec.Properties {
               using System;
                [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTy
pedResourceBuilder", "4.0.0.0")]
               [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
               [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
               internal class Resources {
                  private static global::System.Resources.ResourceManager resourceMan;
                  private static global::System.Globalization.CultureInfo resourceCulture;
                   [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
                  internal Resources() {
                  }
                  [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.E
ditorBrowsableState.Advanced)]
                  internal static global::System.Resources.ResourceManager ResourceManager {
                     get {
                        if (object.ReferenceEquals(resourceMan, null)) {
```

```
                global::System.Resources.ResourceManager temp = new
global::System.Resources.ResourceManager("MultiFaceRec.Properties.Resources",
typeof(Resources).Assembly);
                    resourceMan = temp;
                }
                return resourceMan;
            }
        }
        [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.E
ditorBrowsableState.Advanced)]
        internal static global::System.Globalization.CultureInfo Culture {
            get {
                return resourceCulture;
            }
            set {
                resourceCulture = value;
            }
        }
    }
}
```