МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра програмних систем і технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

Тема: Програмне забезпечення розпізнавання людини у масці

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконавець: Студент групи ІПЗм-21 _____Семен ЯКИМОВИЧ

Керівник: к.т.н., доцент _____Катерина МЕРКУЛОВА

Київ 2022

Рішенням Екзаменаційної комісії

випускна кваліфікаційна робота студента

_____

захищена з оцінкою

_____

Голова комісії

професор д.т.наук Бондарчук Андрій

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій
Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ

Завідувач кафедри

програмних систем і технологій

_____

«_____»_____2022 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ
СТУДЕНТУ

Якимовича Семена Ростиславовича
(прізвище, ім’я, по батькові)

1. Тема випускної кваліфікаційної магістерської роботи: “Програмне забезпечення розпізнавання людини у масці”

затверджена наказом вищого навчального закладу від "__" грудня 2021 року №

2. Строк здачі студентом закінченої роботи: з _____ до _____

3. Вихідні дані до роботи: з використанням відкритих бібліотек та баз даних розробити програмне забезпечення розпізнавання людини у масці

4.Зміст пояснювальної записки (перелік питань, що їх належить розробити):
1) сучасні системи та технології розпізнавання обличь;
2) методи реалізації нейронних мереж для розпізнавання зображень;
3) програмне забезпечення визначення медичної маски на обличчі.

5. Перелік графічного матеріалу (з точним забезпеченням обов’язкових креслень):
1) діаграма прецедентів використання системи;
2) діаграма послідовності дій для запису даних в системі;
3) діаграма класів системи обліку стану здоров’я пілотів авіакомпанії;
4) схема алгоритму попередньої обробки і тренування моделі;
5) схема алгоритму визначення медичних масок на обличчях у зображеннях і у відеофайлах;
6) приклади використання системи розпізнавання людини у масці.

6. Консультанти з роботи із зазначенням розділів роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
I	Меркулова К.В	Меркулова К.В	Якимович С. Р
II	Меркулова К.В	Меркулова К.В	Якимович С. Р
III	Меркулова К.В	Меркулова К.В	Якимович С. Р
IV	Меркулова К.В	Меркулова К.В	Якимович С. Р

7. Дата видачі завдання «___» _____2022 р.

Керівник кваліфікаційної роботи к.т.н., доцент Катерина МЕРКУЛОВА

Завдання прийняв до виконання Семен ЯКИМОВИЧ

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Провести аналіз літератури за темою кваліфікаційної роботи та аналіз існуючих систем		
2	Зробити вибір компонентів системи		
3	Розробити структуру програмних засобів системи		
4	Розробити програмні засоби для розгортання онлайнової системи		
5	Провести налаштування програмних засобів онлайнової системи		
6	Загальне редагування та друк пояснювальної записки, графічного матеріалу		
7	Проходження нормоконтролю		
8	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації		
9	Отримання відгуку керівника і рецензії.		
10	Підготовка матеріалів для передачі секретарю ДЕК		

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи “Програмне забезпечення розпізнавання людини у масці” : 117 с., 49 рис., 28 літературних джерел, 1 додаток.

Мета дослідження – аналіз та вибір методів обробки зображень для створення програмного забезпечення розпізнавання людини у масці.

Об’єкт дослідження – об’єкти на статичних зображеннях, фотографії людей у масці.

Предмет дослідження – методи та алгоритми для розробки системи для розпізнавання облич у масці

Після посилення пандемії *Covid*-19 розробка програмного забезпечення для виявлення медичних масок на обличчі досягло значного прогресу в сферах обробки зображень та комп’ютерного зору. На даний момент вже є багато моделей розпізнавання обличчя, які створені з використанням кількох алгоритмів і методів.

Програмне забезпечення написано мовою *Python* з використанням відкритих для загального використання технічних рішень і фреймворків. Запропонований у роботі підхід використовує глибоке навчання, *TensorFlow*, *Keras* і *OpenCV*. Розроблену модель можна використовувати в цілях безпеки та інформування. Розроблений підхід передбачає виділення окремих зображень з потокового відео або отримання окремих зображень для визначення обличчя з використанням для класифікатора архітектури *MobilenetV*2. Розроблене програмне забезпечення не використовує жодних змінених наборів даних з замаскованими зображеннями, що було обумовлено погіршенням роботи моделі при введенні додаткових навчальних наборів з замаскованими зображеннями.

Базовий набір даних складається з 4100 зображень, які можна розділити на два класи: 2165 зображень з маскою та 1935 зображень без маски. Використані зображення були отримані з наступних відкритих джерел: *Bing Search API, Kaggle, RMFD.*

Досягнуто точність визначення обличь з маскою близько 93%. Було визначено великий вплив якості зображення на визначення обличь з маскою.

НЕЙРОННА МЕРЕЖА, АНАЛІЗ ЗОБРАЖЕНЬ, МАШИННЕ НАВЧАННЯ. ВИЗНАЧЕННЯ ОБ’ЄКТІВ, ОНЛАЙНОВА СИСТЕМА.

ABSTRACT

The explanatory note to the qualification work "A Masked-Face Recognition Software": 118 pp., 49 figures, 28 references, 1 appendix.

The aim of the work is to develop software to ensure the recognition of a person in a mask.

The object of research - recognition of objects on static images.

The subject of research - Software for recognizing a person in a mask.

Following the intensification of the Covid-19 pandemic, the development of software to detect medical masks on the face has made significant progress in the areas of image processing and computer vision. Currently, there are many models of facial recognition, which are created using several algorithms and methods.

The software is written in Python using open source technology solutions and frameworks. The approach proposed in the thesis uses deep learning, TensorFlow, Keras and OpenCV. The developed model can be used for security and information purposes. The developed approach involves selecting individual images from streaming video or obtaining individual images to identify the face using the MobilenetV2 architecture classifier.

The developed software does not use any modified data sets with masked images, which was due to the deterioration of the model when introducing additional training sets with masked images.

The basic data set consists of 4100 images, which can be divided into two classes: 2165 images with a mask and 1935 images without a mask. The images used were obtained from the following open sources: Bing Search API, Kaggle, RMFD.

Achieved accuracy of face detection with mask for images near of the 93%. The great influence of image quality on the definition of masked faces was determined.

NEURAL NETWORK, IMAGE ANALYSIS, MACHINE LEARNING. DEFINITION OF OBJECTS, DESCRIPTION OF THE SYSTEM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ANN – *Artificial Neural Networks*

GPU – *Graphics Processing Unit*

HTTP – *Hyper Text Transfer Protocol*

ISO – *International Organization for Standardization*

NIST – *National Institute of Standards and Technology*

OpenCV – *Open Source Computer Vision Library*

OCR – *Optical Character Recognition*

PNG – *Portable Network Graphics*

ПММ – приховані Марковські моделі

СТЗ – система технічного зору

УДП – умовні довільні поля

ШНМ – штучнима нейронна мережа

ВСТУП

Важливість проблеми розпізнавання образів є однією з найбільш обширних тем теорій інтелектуальних систем. Розпізнавання образів, як проблема, було розглянуто у безлічі наукових праць, а плоди цих робіт мали значуще практичне значення. Розпізнавання являє собою задачу класифікації. Варто відрізняти між собою терміни “класифікація” та “розпізнавання” і розуміти різницю між ними. Хоча терміни і застосовуються у суміжних сферах, їх інтерпретація часто залежить від контектсу використання.

Класифікація представляє собою процесс присвоєння об'єкту певної мітки. Як приклад, розпізнавання дорожніх знаків або задача прийняття рішення про видачу кредиту у автоматизованих банківських системах. Класифікація являє відображає собою задачу віднесення деякої сутності до певного класу. До таких задач, зокрема, належить й розпізнавання образів.

Розпізнавання обличь і об’єктів на обличчі представляє практичну проблему теорії розпізнавання образів. Важливими складовими цієї задачі є ідентифікація облич та їх наступна класифікація за певними ознаками. Ідентифікація особи активно використовується у сучасних сервісах, таких як *Facebook, iPhoto*. Розпізнавання особи використовується повсюдно, починаючи від *FaceID* до *iPhone X*.

Виявлення маски для обличчя виявилося дивовижною проблемою в області обробки зображень і комп’ютерного зору. Розпізнавання обличчя має різні варіанти використання, починаючи від розпізнавання обличчя і закінчуючи фіксацією рухів обличчя, де останнє вимагає розкриття обличчя з дуже високою точністю. У зв’язку з швидким прогресом у сфері алгоритмів машинного навчання, проблеми, пов’язані з технологією виявлення масок для обличчя достатньо добре вирішені. Ця технологія сьогодні є більш актуальною, оскільки

вона використовується для виявлення обличь не тільки на статичних зображеннях і відео, а й в реальному часі. З досягненням згорткових нейронних мереж [1] і глибокого навчання [2], можна досягти дуже високої точності в класифікації зображень і виявлення об’єктів.

Через раптову появу пандемії *COVID*-19 в даний час існують різні технології розпізнавання облич, які застосовуються до людей в масках. *HanvonTechnologyWang* [4] повідомили, що точність розпізнавання обличчя в масці становить близько 85 %. Точність понад 90 % була отримана від *Minivision Technology* [5]. Багатогранна модель на основі обличчя і очей [7] досягає 95% точності розпізнавання. Ван, Лі і Фей (2020) використовували алгоритм *YOLOv*3 для виявлення маски обличчя [8]. Цей метод досяг 93,9 % точності.

Людина розпізнає обличчя інших людей завдяки зоні мозку на межі потиличної та скроневої часток – веретеновидної звивині. Ключові особливості, які виділяє мозок для ідентифікації, – очі, ніс, рот. Людський мозок здатний відновити повну картину об’єкту цілком лише маючи половину потрібних даних. Всі побачені особи мозок усереднює, а потім знаходить відмінність від цього усередненого варіанту. Тому людям європеоїдної раси здається, що всі, хто належить монголоїдній расі схожі один на одного. Представникам китайської держави усі західні люди здаються однаковими. Внутрішнє розпізнавання налаштоване на спектральному діапазоні осіб у голові, тому, якщо якійсь частині спектра не вистачає даних, різні особа на різних зображеннях вважаються за одну.

Завдання щодо розпізнавання осіб вирішують уже понад 40 років. До них входить:

– пошук та розпізнавання кількох осіб у відеопотоці;

–стійкість до змін обличчя, зачіски, бороди, окулярів, віку та повороту особи;

– масштабованість даних для ідентифікації людини;

– робота у реальному часі.

Гістограма спрямованих градієнтів один з оптимальних алгоритмів для знаходження особи на картинці та її виділення. Але незалежно від методів чи підходів до розпізнавання образів проблематика розпізнавання складається з двох частин:

– навчання;

– розпізнавання.

Навчання проводиться шляхом виділення окремих об’єктів з вказуванням їх приналежності тому або іншому образу. В наш час використовується безліч готових інструментів, але і в них нема стовідсотково працюючих методів.

Саме задача виділення або локалізації об’єктів є однією з перших, які розв’язуються при побудові системи розпізнавання образів. Для локалізації об’єктів розроблено десятки методів і алгоритмів, які в основному направлені на аналіз:

– ознак яскравості;

– просторово спектральних ознак;

– контурних ознак.

В останні 50 років набули поширення штучні нейронні мережі (НМ), що зародилися як окрема галузь. Практична цінність теорії НМ є нескінченна. Сьогодні мільярдні інвестиції виділяються для поглиблення знань щодо цих технологій. До проблем, які вирішуються за допомогою НМ відносять: автоматизацію процесів, розпізнавання, апроксимацію, прогнозування, кластеризацію та класифікацію.

Для реалізації локалізації об’єктів за допомогою програми треба зосередитись на алгоритмах та прикладах рішення задач, які зможуть отримати зображення з зовнішнього пристрою або файлу, провести його попередню обробку, локалізувати всі можливі або обумовлені правилами системи групи об’єктів.

З огляду на це можна визначитись з об’єктом, предметом та метою кваліфікаційної роботи.

Мета дослідження – розробка програмного забезпечення для забезпечення розпізнавання людини у масці.

Об’єкт – розпізнавання об’єктів на статичних зображеннях.

Предмет – програмне забезпечення розпізнавання людини у масці.

Результати кваліфікаційної роботи рекомендується використовувати при розробці систем ідентифікації об’єктів з графічних зображеннях.

Впровадження системи виокремлення об’єктів на картинках дозволяє досягти такого ефекту внаслідок:

1) автоматизація процесу ідентифікації об’єктів на зображенні;

2) скорочення часу на попередню обробку зображень для подальшого аналізу;

3) можливості використання даних методів і алгоритмів в системах відеоспостереження або при потоковій роботі з відеорядом.

РОЗДІЛ 1
СУЧАСНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ОБЛИЧЬ

Середньостатистична людина може ідентифікувати знайому особу в натовпі з точністю 97,53%. Але у порівнянні із сучасними алгоритмами, які досягли точності 99,8% ще у 2014 році, це вже замало. Сучасні алгоритми, що використовуються в камерах відеоспостереження в багатьох європейських і американських містах, здатні обробляти 1 мільярд зображень менш ніж за півсекунди з точністю близькою до 100%.

Виявлення маски для обличчя виявилося дивовижною проблемою в області обробки зображень і комп'ютерного зору. Розпізнавання обличчя має різні варіанти використання, починаючи від розпізнавання обличчя і закінчуючи фіксацією рухів обличчя, де останнє вимагає розкриття обличчя з дуже високою точністю. У зв'язку з швидким прогресом у сфері алгоритмів машинного навчання, проблеми, пов'язані з технологією виявлення масок для обличчя достатньо добре вирішені. Ця технологія сьогодні є більш актуальною, оскільки вона використовується для виявлення обличь не тільки на статичних зображеннях і відео, а й в реальному часі.

Через раптову появу пандемії *COVID*-19 в даний час існують різні технології розпізнавання облич, які застосовуються до людей в масках. *HanvonTechnologyWang* [4] повідомили, що точність розпізнавання обличчя в масці становить близько 85 %. Точність понад 90 % була отримана від *Minivision Technology* [5]. Багатогранна модель на основі обличчя і очей [7] досягає 95% точності розпізнавання. Ван, Лі і Фей (2020) використовували алгоритм *YOLOv*3 для виявлення маски обличчя [8]. Цей метод досяг 93,9 % точності.

На даний момент доволі багато готових відкритих для використання, які включають алгоритми розпізнавання образів. Список популярних бібліотек, які використовуються в сучасних системах:

– *DLIB*;

– *OpenCV*;

– *iOS Vision Framework*.

Кожна з систем має свої переваги та недоліки.

1) бібліотека *DLIB*:

Переваги:

– *open source* рішення, можна брати участь у розвитку і дивитися поточні тренди;

– написана на C++. Має підтримку для *iOS* як *cocoapods*: *pod* '*dlib*';

– можна також інтегрувати як *C++* бібліотеки. Працює на *Windows, Linux, MacOS*;

Недоліки:

– великий розмір бібліотеки, що підключається. 40 мегабайт як *pod*;

– високий поріг входу. Велика кількість внутрішніх алгоритмів, під кожен із яких належить писати обгортку на *Objective-C*;

2) бібліотека *OpenCV* (рис. 1.1):

Переваги:

- найбільше ком'юніті, що регулярно бере участь у підтримці.

- написана на C++. Має підтримку для *iOS* як *cocoapods*: *pod* '*OpenCV*'.

Недоліки:

– високий поріг входу;

– великий розмір бібліотеки, що підключається. 77 мегабайт як *pod*, 180 мегабайт як *C++* бібліотеки.

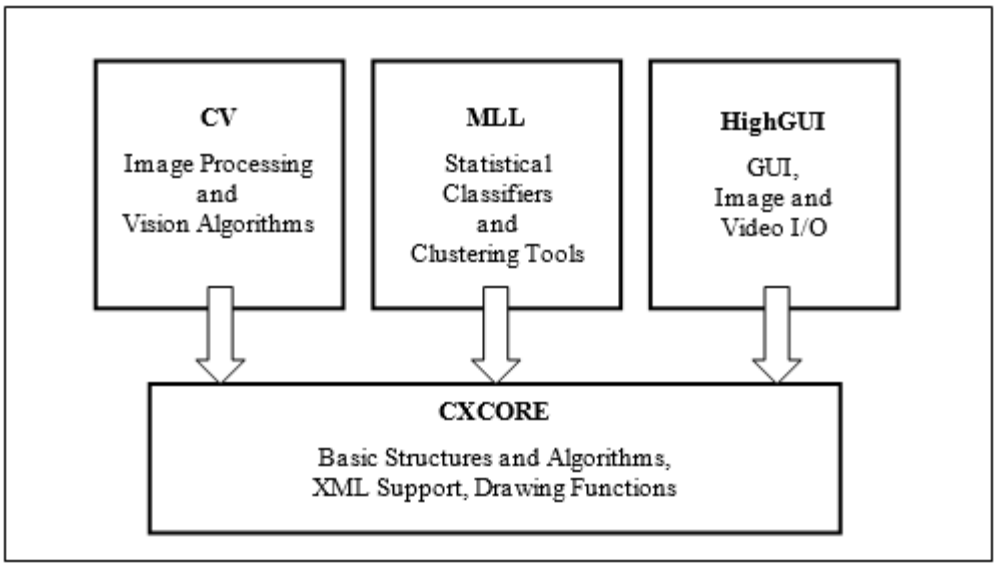


Рис. 1.1. Структура бібліотеки *OpenCV*

3) *iOS Vision Framework*

Переваги:

– проста інтеграція в додаток;

– підтримує фреймворки *Keras, Caffe, scikit-learn;*

– рішення з малим розміром;

– працює на *GPU*.

Недоліки:

– є частиною *CoreML* (рис. 1.2);

– немає підтримки *TensorFlow*, одного з найпопулярніших рішень машинного навчання. Доведеться витрачати багато часу на самописні конвертери;

– є високорівневою абстракцією. Вся імплементація закрита, звідси неможливість контролю. *iOS* 11+.

Існують платні платформи, які надають рішення для розпізнавання образів. Більшість розвиває власні алгоритми та технології. Звісно ж, ці технології активно розвиваються та використовуються військовими, тому деякі рішення засекречені і не мають відкритих вихідних джерел.

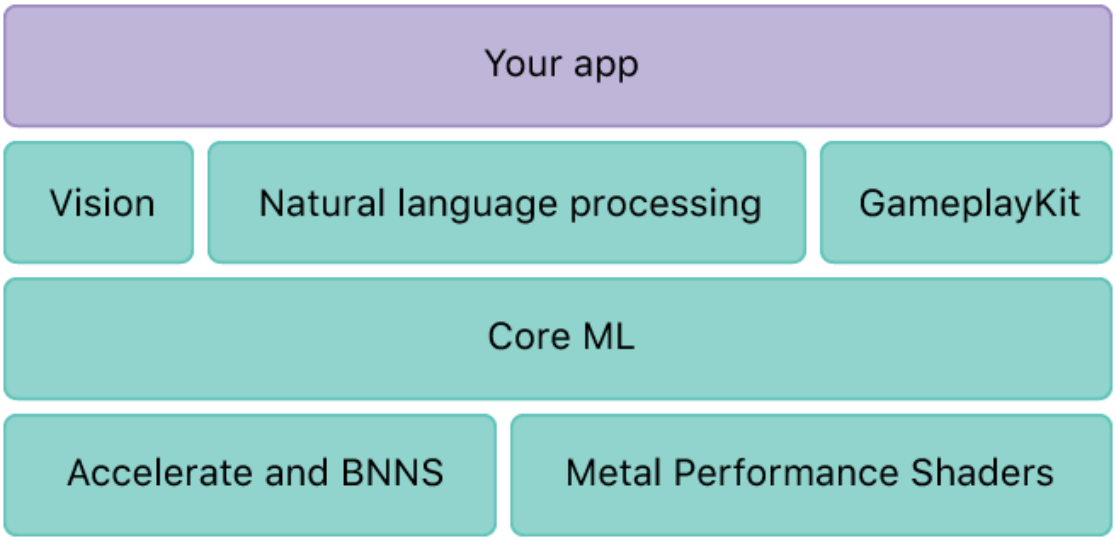


Рис 1.2. Структура *CoreML*

У відповідності до матеріалів робіт [2, 10, 11] знаходження точок особи починається з пошуку *landmarks* – знаходження точок особи. Перший крок у

алгоритмі - визначення локації особи на картинці. Після отримання локації особи шукають ключові контури:

– контур особи;

– ліве око;

– праве око;

– ліва брова;

– права брова;

– ліва зіниця;

– правий зіниця;

– ніс;

– губи.

Кожен із цих контурів є масивом точок на площині (рис. 1.3).

На малюнку можна чітко побачити структури обличчя. При цьому, залежно від обраної бібліотеки, кількість *landmarks* відрізняється.

Більш потужним рішенням є тріангуляція Делоне для побудови маски (рис. 1.4).

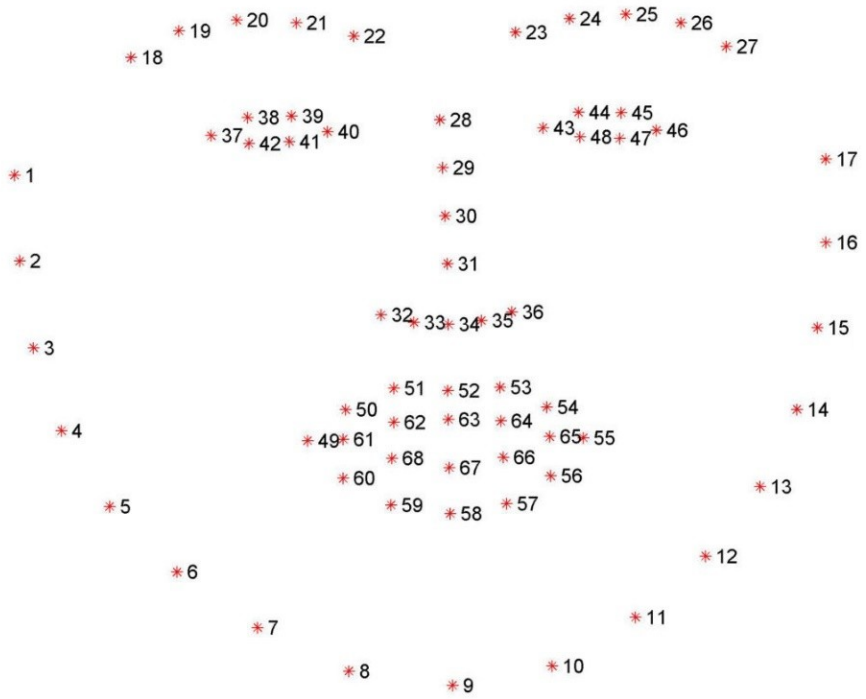


Рис. 1.3. Приклад визначення *landmarks* в *DLIB*

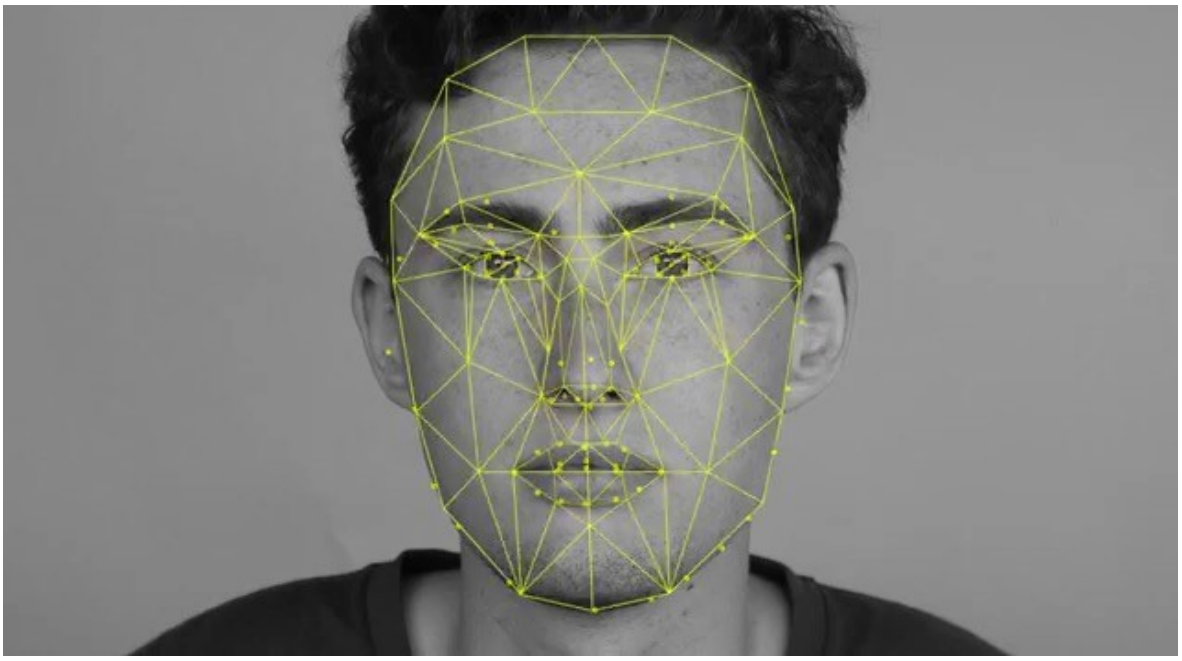


Рис. 1.4. Маска, що візуалізує алгоритм тріангуляції Делоне

Тріангуляція Делоне – тріангуляція для множини точок *S* на площині, при якій для будь-якого трикутника всі точки з *S* за винятком точок, що є його вершинами, лежать поза колом, описаним навколо трикутника. Вперше описана 1934 року радянським математиком Борисом Делоне (рис. 1.5).

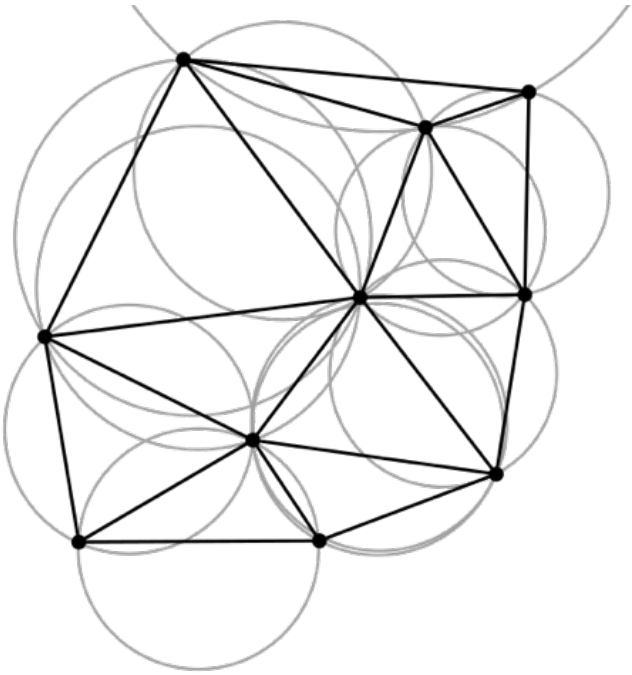


Рис. 1.5. Приклад тріангуляції Делоне

1.1. Технології машинного зору та розпізнавання осіб

Технології машинного зору та розпізнавання осіб розвивалися дуже активно із середини минулого століття [9, 10, 12, 14]. Але лише зараз стали по-справжньому добре працювати. Цьому факту є три основні причини:

– з’явилися справді потужні комп'ютери, здатні впоратися із завданням;

– з’явилися бази даних із колекціями фотографій;

– прорив у галузі нейромереж.

Усі ці події дозволили створити практично ідеальні алгоритми розпізнавання обличь.

Всі алгоритми розпізнавання проходять подібні один до одного етапи на шляху до розпізнавання:

– етап 1. Виявлення зони обличчя;

– етап 2. Антропометричні точки;

– етап 3. Виправлення спотворень;

– етап 4. Вектор обличчя.

Щоб обличчя розпізнати, треба його спочатку виявити. Для цього можна було б використовувати натреновані нейромережі, але це занадто довго, дорого і ресурсомістко. Тому в роботі для виявлення обличчя використовується метод Віоли-Джонса, розроблений у 2001 році.

Цей алгоритм просто сканує зображення за допомогою прямокутників (рис. 1.6), вони називаються примітивами Хаара:

Завдання цих об'єктів – шукати світлі і чорні області на зображенні, властивих саме людським обличчям.

Наприклад, якщо усереднити значення яскравості область очей буде темніша за щіку або чоло, а перенісся буде світлішим за брів (рис. 1.7).

Загалом таких характерних ознак багато й природно у людських осіб може бути подібні патерни. Тому алгоритм працює у кілька етапів. Спочатку знаходиться перша ознака, система розуміє: «У цій зоні може бути особа». Тоді вона починає там же шукати другу ознаку, а потім третю. І якщо в одній області знайдено три ознаки, вже можна впевнено сказати, що це обличчя.

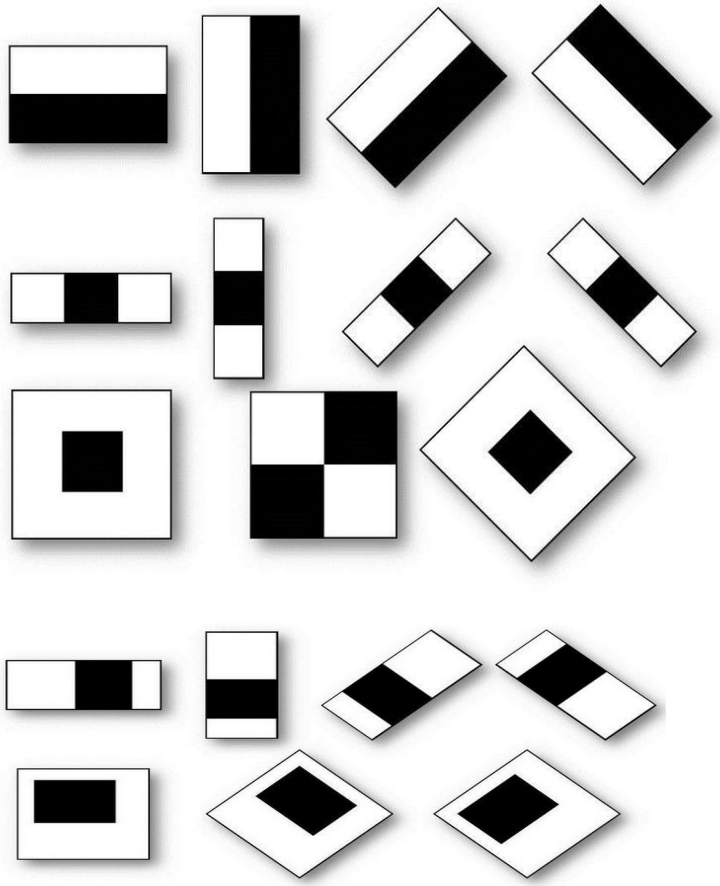


Рис. 1.6. Примітиви Хаара

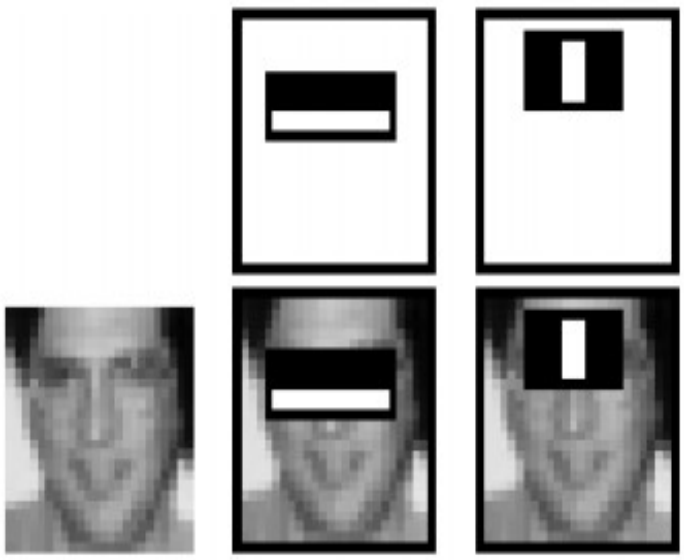


Рис. 1.7. Приклад пошуку ознак обличчя за допомогою примітивів Хаара

Отримавши область для аналізу, далі в дію вступає головний секрет кожної системи розпізнавання – біометричний алгоритм. Він розставляє на обличчі антропометричні точки, за якими згодом і обчислюватимуться індивідуальні

характеристики людини: відстань поміж очами, вигляд носа, загальне розміщення та інше. Таких ознак може бути багато, аж до кількох тисяч [7]. Але загалом таких точок має бути як мінімум 68 (рис.1.8).

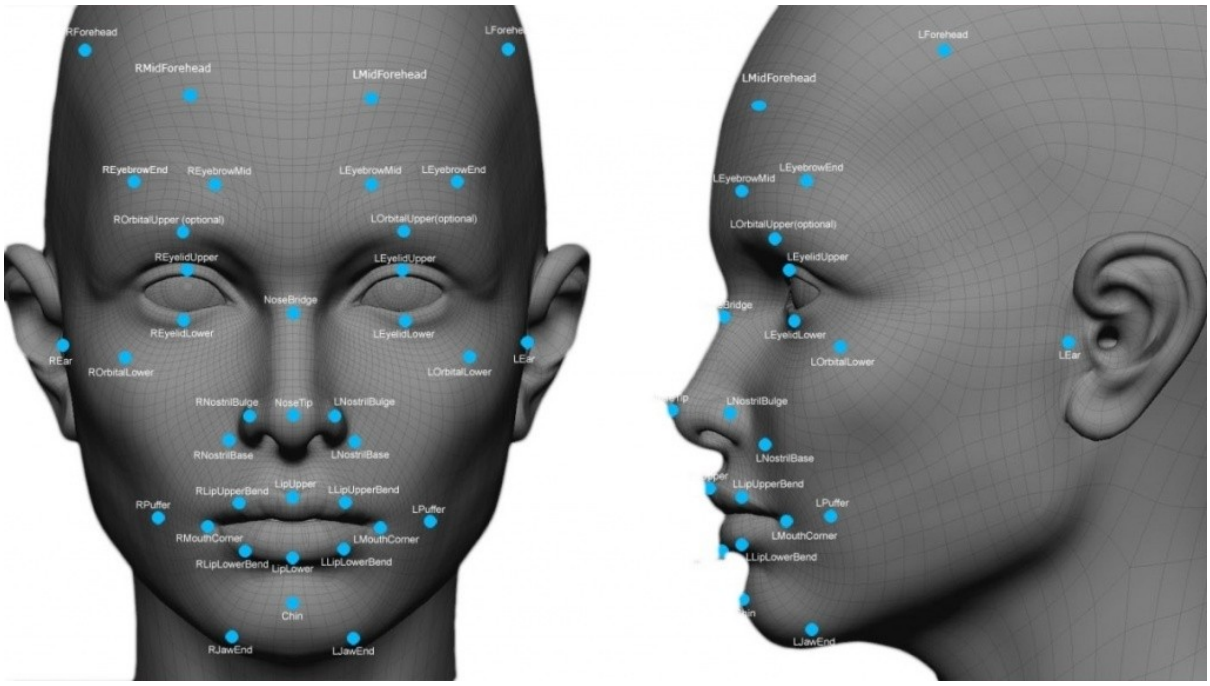


Рис. 1.8. Приклад антропометричних точок

Далі система здійснює додаткове перетворення зображення: усунуться поворот і нахил голови (рис. 1.9). А також проводиться 3*D*-реконструкція обличчя із 2*D*-зображення. Таким чином, навіть якщо людина на зображенні дивилася вбік, все одно можемо отримати чіткий фронтальний знімок, що істотно підвищує якість розпізнавання.

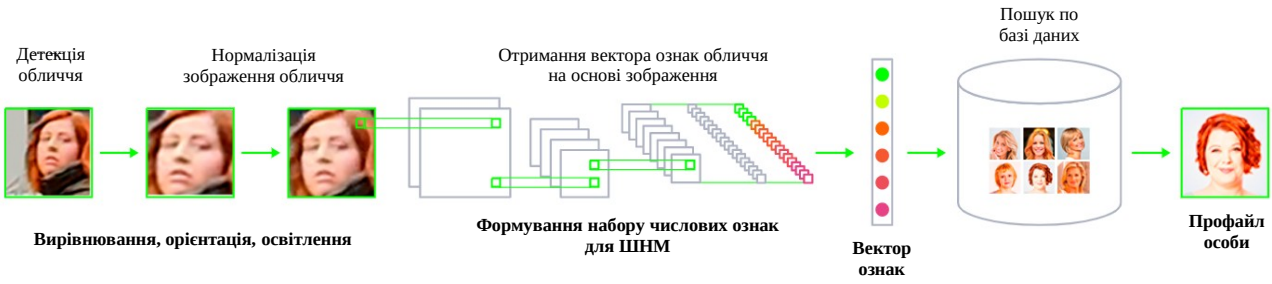


Рис. 1.9. Етапи визначення особи на основі зображення обличчя

Далі для подачі даних на вхід нейромережі формується вектор ознак. По суті, це число, яке складається із суми характеристик особи: відстаней між опорними точками, текстури певних областей на обличчі та інше. Таких

параметрів може бути безліч. Основне правило: вони повинні описувати особу незалежно від сторонніх факторів: макіяжу, зачіски, вікових змін.

Після чого залишається тільки порівняти отриманий вектор із базою інших векторів.

1.2. Описання принципів семантичної сегментації

1.2.1. Класичні методи

Семантична сегментація зображення означає присвоєння кожному пікселю певної позначки [6, 7, 12]. У цьому полягає головна відмінність від класифікації, де всьому зображенню ставиться у відповідність лише одна мітка. Сегментація працює з множиною об'єктів одного класу як із єдиним цілим.

Інстанс-сегментація обробляє кілька об'єктів одного класу як різноманітні об'єкти. Зазвичай інстанс-сегментація складніша, ніж семантична сегментація (рис. 1.10).

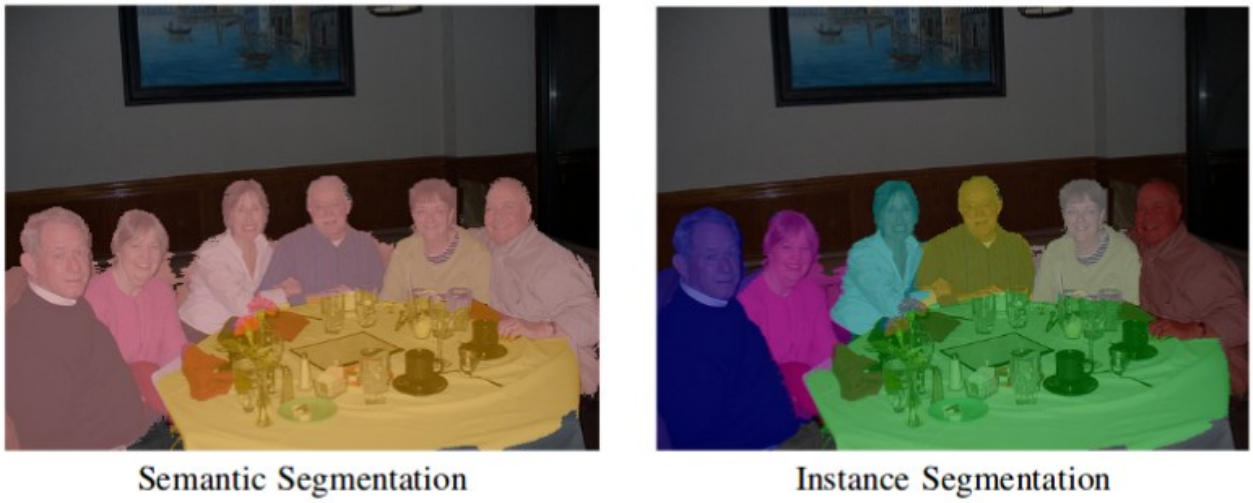


Рис. 1.10. Порівняння інстанс- та семантичної сегментації

1.2.1.1. Сегментація з використанням градації сірого

Сегментація зі застосуванням градації сірого дозволяє за допомогою механічних ознак приписати та надати об’єкту певну мітку. Ці правила можуть бути оформлені як характеристики пікселів, наприклад, інтенсивність сірого кольору. Приклад такого виступає алгоритм поділу та об’єднання (Split and

Merge). За допомогою принципу рекурсії, виконується розділ зображення на декілька областей, кожні з яких кластеризуються. Суміжні області з одного кластеру об’єднуються.

Алгоритм не є автоматичним, тобто принципи розділення задаються вручну. На превеликий жаль, складні області та об’єкти, такі як «очі», може бути не завжди можливо відрізнити лише за допомогою вторинних сірих ознак. Дуже часто такий алгоритм поєднується з іншими системами для фільтрації та попередньої обробки даних.

1.2.1.2. Умовні випадкові поля

У випадку, якщо модель не ідеальна, можна отримати результати із зашумленою сегментацією, що часто неможливо в природі (наприклад, на рис. 1.11 пікселі котиків змішуються з пікселями собак, як показано на зображенні).

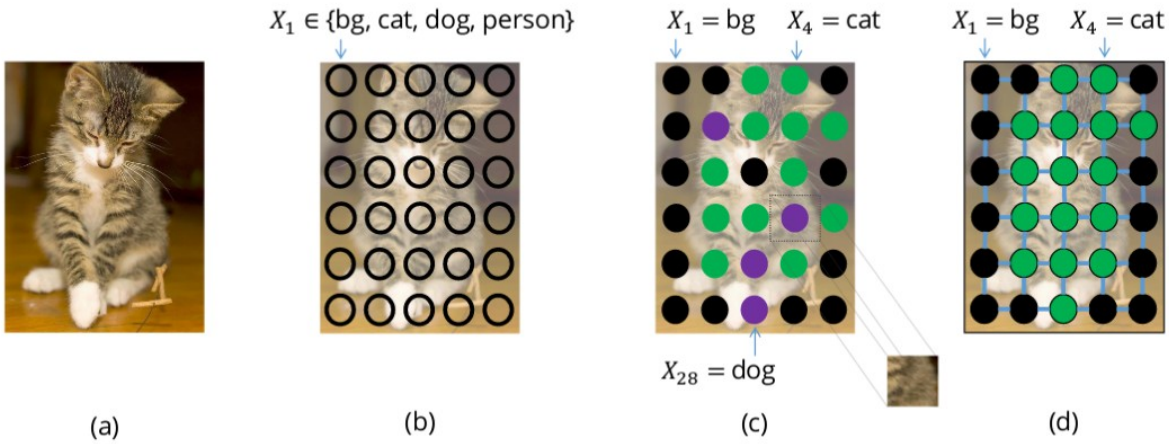


Рис. 1.11. Приклад зашумленої сегментації: *a*) зображення, *b*) сегментація, *c*) пікселі з мітками собак змішані з мітками котів, *d*) більш природна сегментація

Цього можна уникнути, якщо розглянути попередні зв'язки між областями. Алгоритм базується на використанні пікселів безперервного кольору для їхнього подальшого відокремлення та присвоєння їх до однієї групи. Для такого правильно буде використовувати УДП, що розшифровується як умовні довільні поля.

УДП відносяться до класу методів статистичного моделювання, що використовуються для структурованих прогнозів. На відміну від класифікаторів перед передбаченням УДП беруть до уваги контекст.

Кожен піксель зображення асоціюється із кінцевим набором можливих станів. На рис. 1.12 стани описуються можливими мітками. Також можна відокремити попарні витрати, які характеризуються відношення пари міток до пари пікселів. Пари пікселів у цьому випадку є сусідами. Також можливо працювати з усіма парами пікселів зображення (щільні УДП).

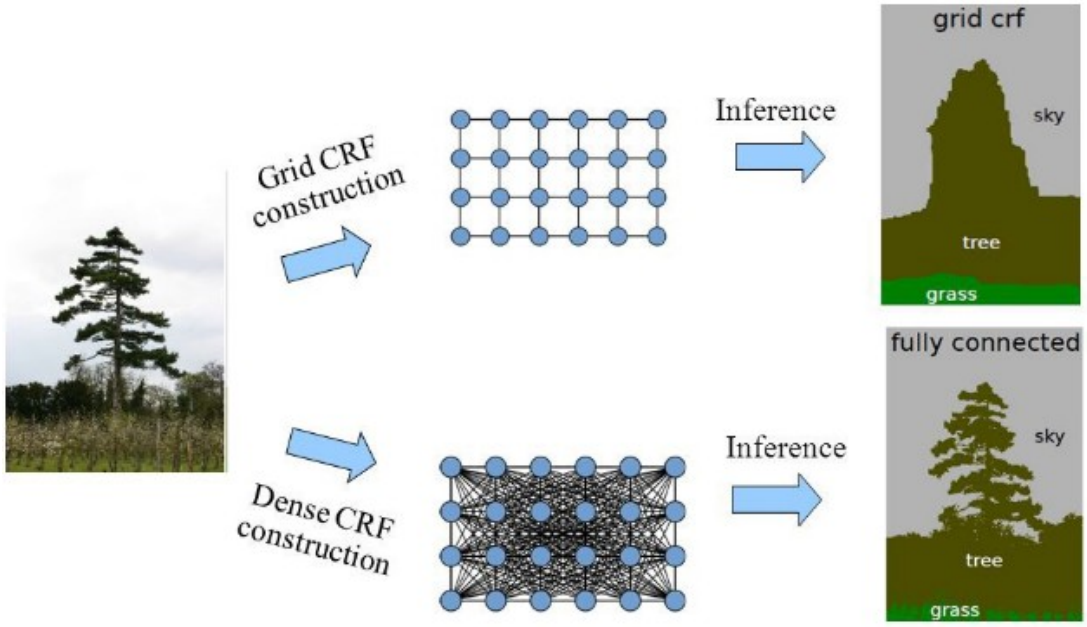


Рис. 1.12. Порівняння сіткових (*Grid CRF*) та щільних (*Dense CRF*) УДП

Сума унарних та попарних витрат на всі пікселі має назву енергії (або витрат/втрат) УДП, яке необхідно мінімізувати.

1.2.2. Аналіз методи глибокого навчання

Глибоке навчання багато в чому спростило пайплайн виконання семантичної сегментації, водночас показавши вражаючу якість. Повнозгорткова мережа (*Fully Convolutional Network, FCN*) представляється як загальноприйнята архітектура для семантичної сегментації. *FCN* використовують для проєкції вхідного зображення на меншу площину за допомогою згорток. Таке перетворення може сприйматися як кодування. Пізніше, прогнозується результат роботи системи за допомогою спеціальної системи декодингу.

Звичайно, така система не позбавлена недоліків. Один з яких пов’язаний з наявністю внутрішніх дефектів формату вводу даних та відповідне прогнозування виходу на основі вхідних даних.

Для збільшення якості базової моделі *FCN* було запропоновано кілька рішень. Нижче наведено деякі з рішень, які довели свою ефективність:

1.2.2.1. *U-Net*

Мережа *U-Net* являє собою вдосконалення *FCN* архітектури. S*kip*-зв'язок складається з блоків згортки та входами блоку згортки на тому ж рівні.

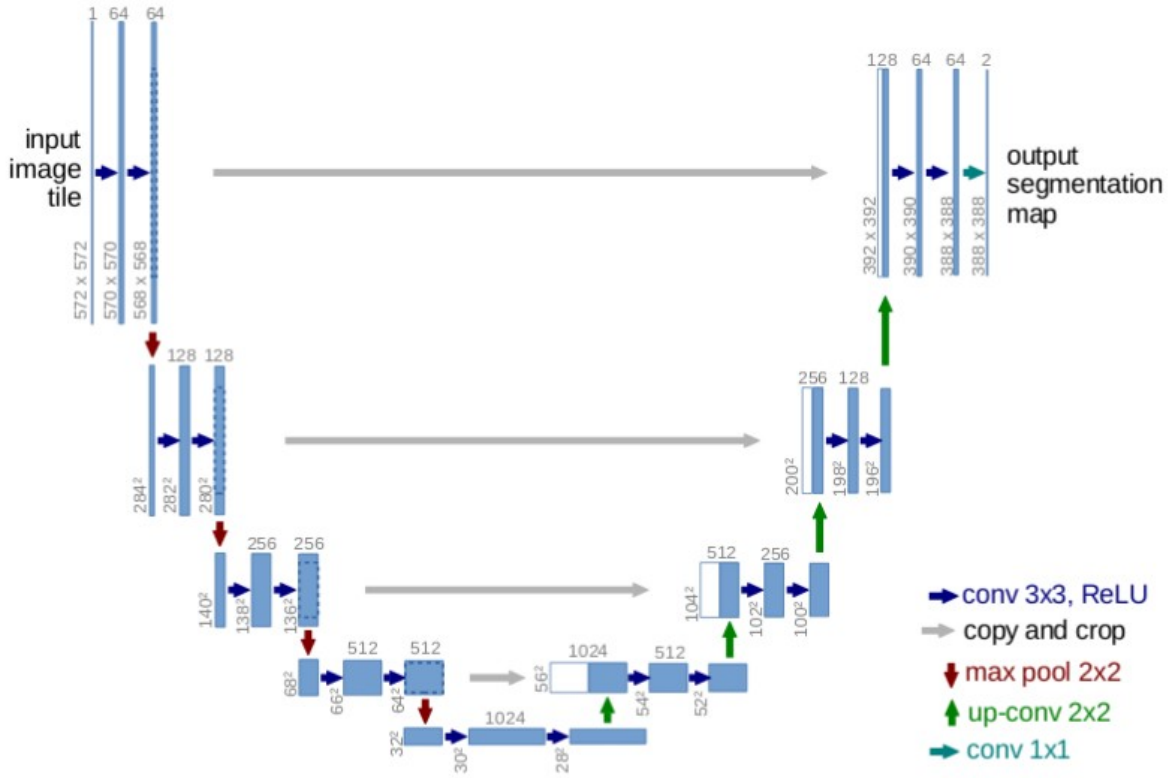


Рис. 1.13. Схема *U-Net* мережі

Skip-зв'язки дозволяють градієнтам краще поширюватися та краще представляти інформацію з різних частин зображення. Модель може краще класифікувати як з малих, так і з великих масштабів. Інформація з менших масштабів використовується для покращення існуючої розділення зображення на сегменти.

1.2.2.2. Модель тірамісу

Являє вдосконалену модель *U-Net* за винятком того, що для прямої та транспонованої згортки тут використовуються щільні блоки. Декілька рядів згорткових шарів нейромережі передають наступним рядам інформацію щодо їх стану. Результуюча мережа надзвичайно ефективна в сенсі параметрів і може працювати з ознаками зі старих шарів.

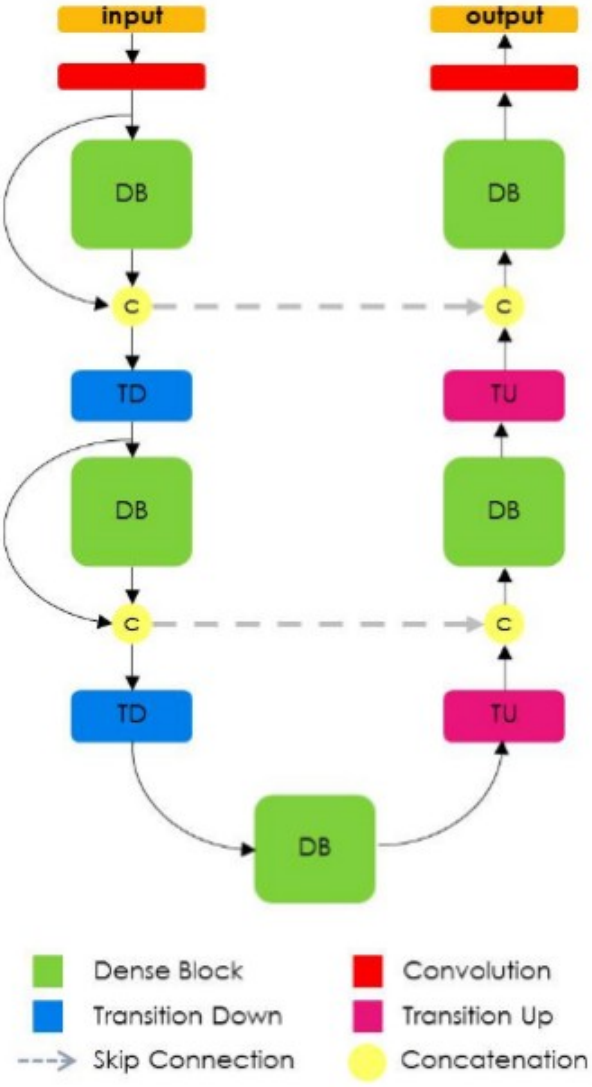


Рис. 1.14. Схема мережі Тірамісу

Кількість витраченої пам’яті та черезмірне використання комп’ютерних можливостей описують недолік такого методу. Варто забезпечитися потужними системами з використанням *GPU* для роботи з такими мережами.

1.2.2.3. Багатомасштабний метод

Деякі методики глибокого навчання базуються на явному перетворенні даних, зокрема їх представленням у різних масштабів. Наприклад, *Pyramid Scene Parsing* (*PSPNet*) використовує безліч внутрішніх перетворень, серед яких об’єднання (через функцію *average*) за допомогою ядр різної розмірності та у декілька ітерацій, застосованими до вихідних попереднії перетворень ознак зображення як за допомогою нейронної мережі, так і шляхом ручного очищення зображення. Опціонально може використовуватись білінійна інтерполяція для обчислення виходів з проміжного шару та відображення ознак; Пізніше, йде

об’єднання виходів системи у один результат. Для створення прогнозу фінальна згортка виконується вже на об'єднаному результаті.

Модель згорток *Atrous Convolutions* є одним із найбільш прийнятих та застосованих способом комбінування попередніх показників з різних розширень з дотриманням оптимальної кількості ознак та змінних. За допомогою мікроналаштувань, можна коригувати ваги далі в просторі. Це дозволяє вивчати більш загальний контекст.

1.2.2.4. Гібридні методи

Можливе комбінування методів для отримання первинних ознак, які вже пізніше передаються до УДП. Такий гібридний метод показує хороші результати через здатність УДП моделювати зв'язок між пікселями.

Деякі методи використовують УДП у самій нейронній мережі, як це зроблено в роботі [6], де використовуються рекурентні нейронні мережі (*RNN*). Таким чином досягається більша ефективність нейронних мереж для вирішення задачі при зменшених затратах ресурсів.

1.3. Висновки до розділу 1

Даний розділ показав, що практично всі алгоритми розпізнавання проходять подібні один до одного етапи на шляху до розпізнавання:

1) виявлення зони обличчя;

2) визначення антропометричних точок;

3) виправлення спотворень;

4) формування вектор обличчя.

Виявлення зон обличчя та антропометричних точок передбачає використання алгоритмів сегментації Технології глибокого навчання суттєво покращили та спростили алгоритми семантичної сегментації, проклавши шлях для більш широкого застосування у реальному житті.

РОЗДІЛ 2
МЕТОДИ РЕАЛІЗАЦІЇ НЕЙРОННИХ МЕРЕЖ ДЛЯ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

2.1. Методи поглибленого навчання нейронних мереж

Нейронні мережі – це основа глибокого навчання. Такі системи глибинного навчання можуть розглядатися як чорні скриньки, у яких невідомо які дії виконуються, лише отримується вихід на основі входу.

Виходи елементів мережі, нейронів, обчислюються за домопогою поняття зваженої суми $\sum_j w_j x_j$. Важливо також встановити коректне порогове значення. Ваги та поріг – дійсні числа та представляють мікропараметри мережі. Якщо розглядати математичні формули, то:

$$output = \begin{cases} 0, \text{ якщо } \sum_j w_j x_j \leq \delta \\ 1, \text{ якщо } \sum_j w_j x_j > \delta \end{cases},$$

де δ – порогове значення.

Така модель отримала назву Перцептрон — формули для прийняття рішень за допомогою наявної інформації

Починаючи зліва, маємо:

– вхідний шар нашої моделі оранжевим кольором;

– перший прихований шар нейронів блакитного кольору;

– другий прихований шар нейронів у пурпуровому кольорі;

– прогноз моделі.

Стрілки показують взаємозв’язок нейронів та пропагацію від вхідного шару аж до вихідного шару.

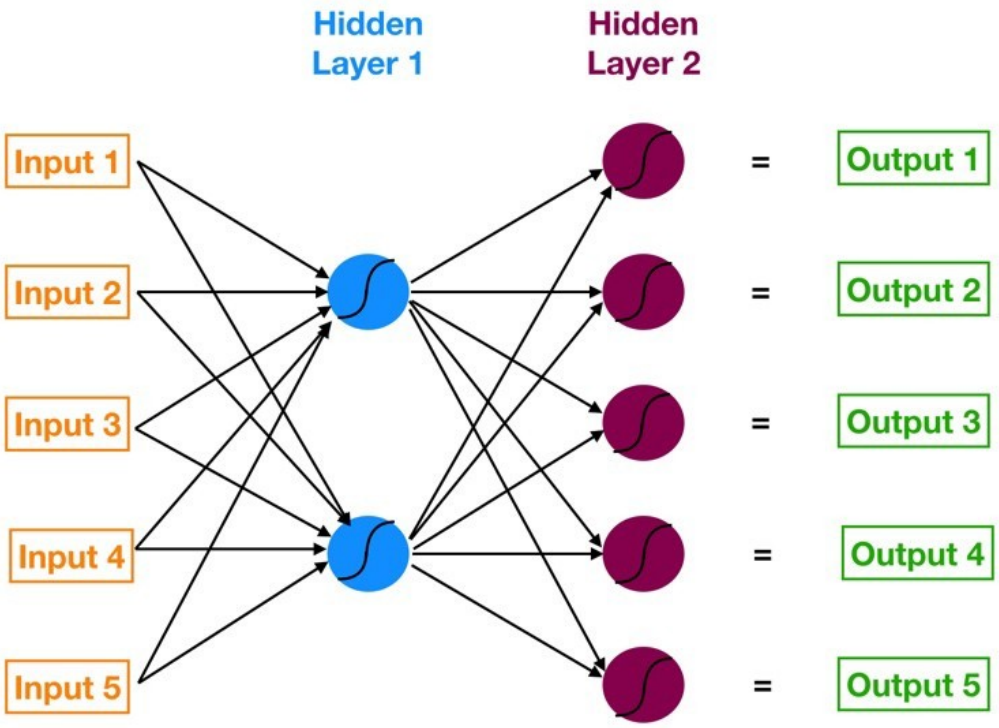


Рис. 2.1. Нейромережа з двома прихованими шарами

Пізніше будемо покроково обчислювати кожне вихідне значення. Нейромережа здатна вчитися за допомогою алгоритму Зворотне розповсюдження, або BackPropagation.

Цей алгоритм поступово покращує існуючий прогноз нейронної мережі. За допомогою обчислення різниці цільових значень та значень отриманих від вхідних даних, мережа вносить коригування у свою структуру.

Таким чином з часом тренування досягається найкращий результат для певного набору даних.

Така система є реалізацією логістичної регресії за допомогою використання нейронних мереж.

Таким чином логістична регресія може бути повністю виражена через нейронну мережу. Для зображення деталей роботи алгоритму, використовуються різні кольори у елементах нейронної мережі. Таким чином, можна побачити взаємозв’язок між ними.

2.1.1. Рівняння логістичної регресії

Розглянемо кожен елемент логістичної регресії:

$$Sigmoid\ (B_1 \cdot X + B_0) = PredictedProbality$$

де X – це вхід, одинока особливість, яку надаємо нашій моделі для обчислення прогнозу.

B_1 – параметр нахилу логістичної регресії – B_1 показує наскільки змінюються параметр “лог оддс“ при зміні X. Варто зазначити, що B_1 знаходиться на лінії, яка поєднує вхід X із синім нейроном у шарі 1.

B_0 – це упередження – дуже схоже на термін перехоплення від регресії. Ключова відмінність полягає в тому, що в нейронних мережах кожен нейрон має власний термін зміщення (тоді як при регресії модель має особливий термін перехоплення).

Функція сигмоїдної активації є невідмінною частиною перцептрону. Така функція відіграє важливу роль у роботі нейрону. Пам’ятайте, що саме використовуємо сигмоподібну функцію перейти від *log-odds* до ймовірності.

Нарешті, можна отримати передбачувану імовірність, за допомогою сигмоїдних функцій.

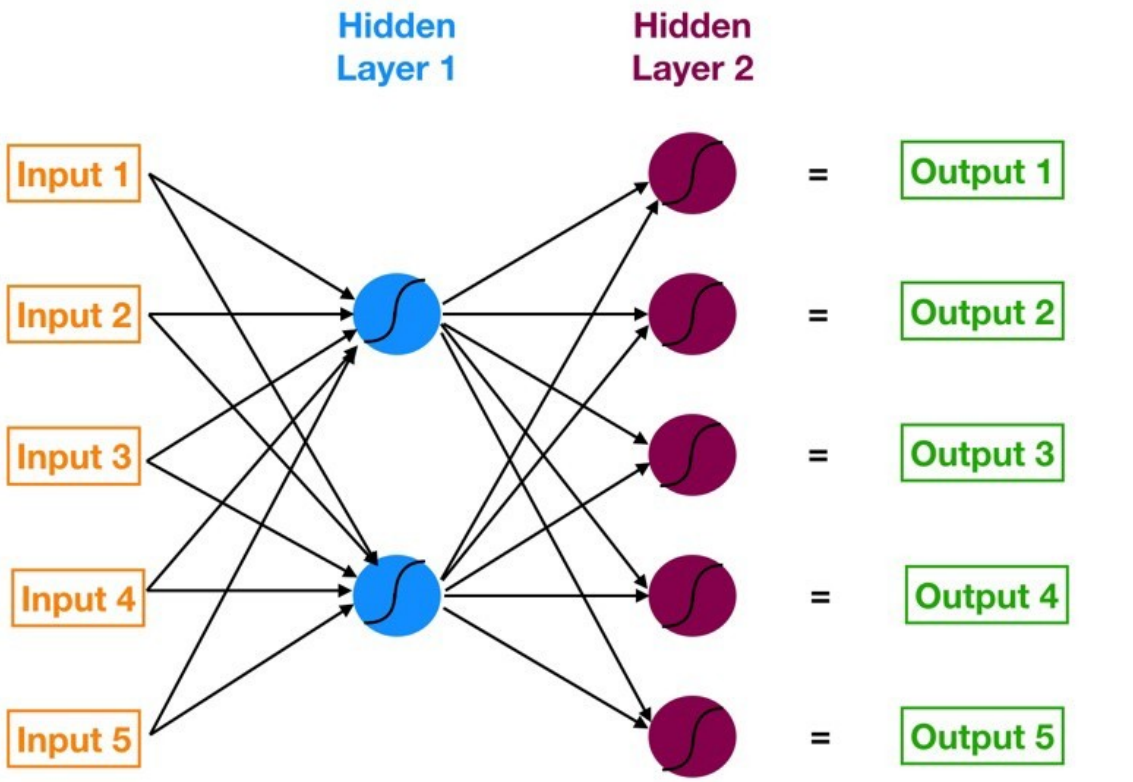
Варто зробити висновок. Нейронна мережа складається тільки й тільки з наступних елементів:

Зв'язок та з’єднання з вагами поміж нейронами які перетворюються у вхід та передають його до наступних нейронів по черзі.

Нейрон, що складається з відступу B_0 та функцію активації, яка може бути лінійна або сигмоїдна.

Саме на цих інструментах будуються найскладніші нейромережі у світі. Ще складніші нейронні мережі це моделі з більшою кількістю нейронів та більшими прихованими шарами. Таким чином можуть створюватися надликові зв’язки. Таке розташування та використання зв’язків може як покращити нейромережу у цілому, так і погіршити її якість.

Ускладнюємо задачу подачі нейронної мережі. Варто розглянути основну нейромережу. Для цього потрібну переглянути складнішу нейронну мережу та подивитись її роботу від входу до виводу. Така мережа зображена нижче:



\

Рис. 2.2. Ускладнена нейронна мережа

Перший шар номер один складається із двох нейронів. Для паралельного підключення треба використати 10 взаємозв’язків. Зображення нижче відображає зв’язки між різними іншими елементами нейромережі та шаром один.

Варто акцентувати на позначенні “маси” та “пріорітету”. За допомогою літери W та двох чисел позначається напрям роботи ваги. Загальне позначення буде виглядати у вигляді запису W, a, b , де a позначає свій власний взаємозв’язок між нейроном b.

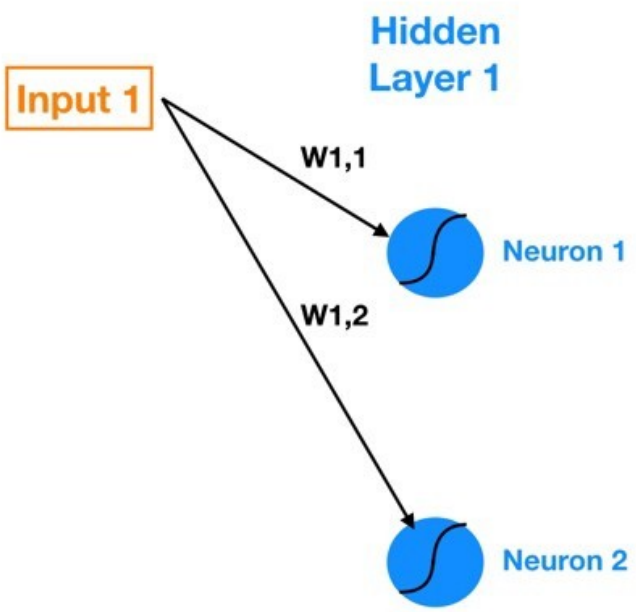


Рис. 2.4. Зв'язки між входом 1 і прихованим шаром 1

Надалі треба розрахувати результати роботи нейронів в прихованому шарі під номером один. Це можна зробити використовуючи наступні формули (*W* позначає вагу, *In* позначає введення).

$$Z_1 = W_1 \cdot In_1 + W_2 \cdot In_2 + W_3 \cdot In_3 + W_4 \cdot In_4 + W_5 \cdot In_5 + \text{зміщення}$$

Де Z = Сигмовидний Z_1.

Для таких обчислень у системах може використовуватися спеціальні інструменти та засоби прискорених матричних обчислень На практиці матричні обчислення ефективно справляються із задачею.

2.1.2. Матрична передставлення нейронної мережі

Будь-яку нейронну мережу можна представити як матрицю, а саме матрицю m елементів глибиною попереднього, та n елементів глибиною наступного шарів:

$$[W] @ [X] + [\text{Упередження}] = [Z]$$

де W –матриця ваг n на m, X – матриця m на 1, *Упередження* – це n на 1 матриця упереджень,

Z – n на 1 матриця виходів.

@ – позначення множення матриць.

Отримавши *Z*, можемо використати функцію активації лінійну або будь-яку іншу до кожного елемента *Z*. Таким чином ми обчислили активації для поточного шару (рис. 2.5).

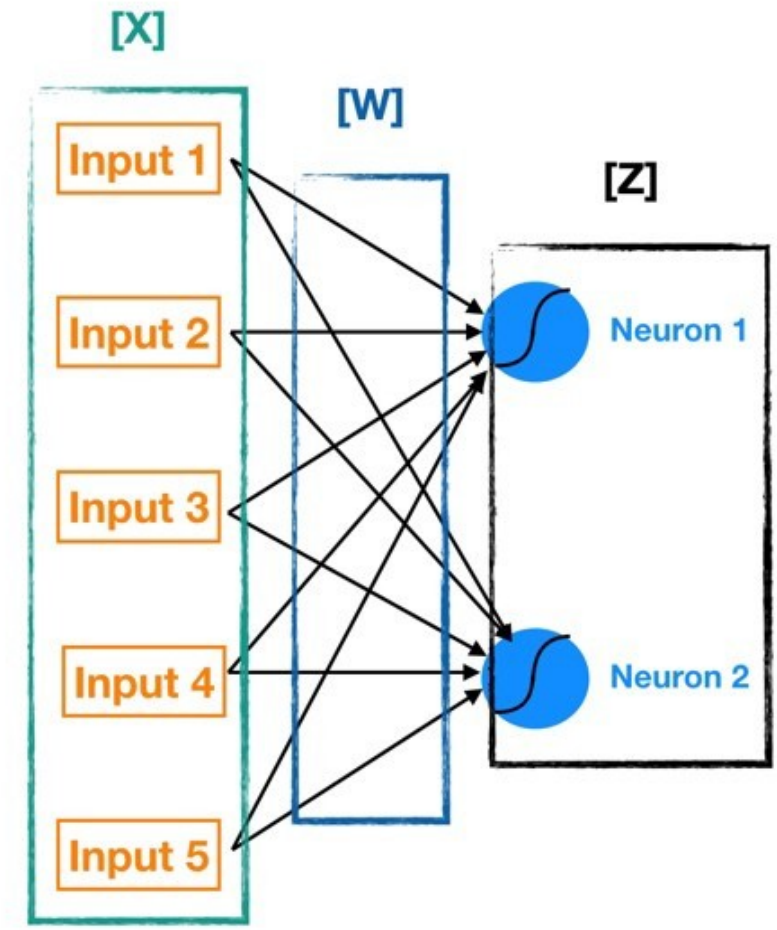


Рис. 2.5. Візуалізація *W*, *X* та *Z*

Обчислюючи *Z* за допомогою наступного використання функції активації для кожного шару, можна поступово переходити між шарами. Такий процес має назву пряма пропагаці. Після цього обчислюються результати які можна використовувати для аналізу якості навчання нейронної мережі.

2.2. Навчення нейронної мережі

Навчальний процес нейронної мережі дуже подібний до точних математичних розрахунків у інших науках – визначається функція витрат та використовується оптимізація градієнтного спуску, щоб її мінімізувати.

Варто провести детальний аналіз та визначити що саме впливає на якість результату нейронної мережі. Загалом, задача зводиться до пошуку коефіцієнтів рівнянь, підставивши які можна отримати найкращий результат.

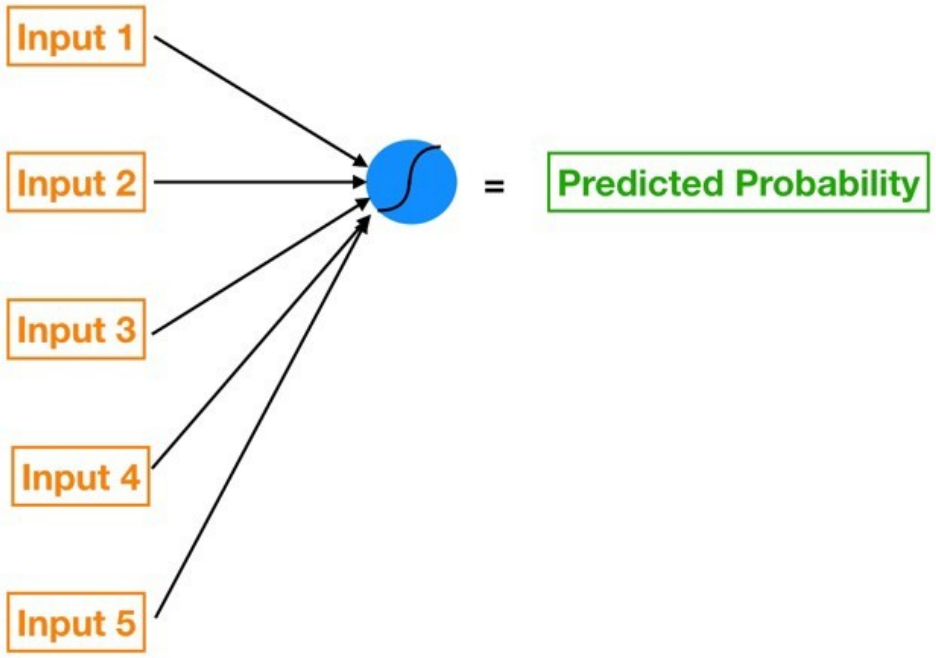


Рис. 2.6. Представлення логістичної регресії

Під час традиційної регресії можна регулювати один коефіцієнт незалежно від іншого. Таким чином можна подивитись вплив саме цього числового значення на весь результат роботи системи. Певні коливання саме цього числового показника можуть значно вплинути як на сусідні параметри так і на мережу у цілому.

Варто виокремлити різні особливості логістичної регресії, реалізованої за допомогою нейронної мережі

У нейронній мережі найменша зміна параметрів будь-якого з’єднання може відобразити зворотній ефект на всі інші нейрони та їх активацію у наступних слоях.

Кожен нейрон схожий за своєю сутністю на мікросхему. Кожен нейрон таким чином представляє собою маленьку модель для навчання. Певні лінійні розподіли елементів можна виразити лише за допомогою використання одного нейрону.

Таким чином, виходить що власне прихований шар представляється як набір мікроскопічних моделей, інформація по яким іде впорядковано та по строгому напряямку. Кожний шар може містити різну кількість нейронів, деякі більше, деякі меньше.

Задача полягає у знаходженні числових коєфіцієнтів ваг, матриці, що представляє з’єднання елементів нейромережі. Також задачі властиве знаходження оптимальних функцій активацій та їх порогових значень. Лише при отримані вдалого результату можна вважати задачу вирішенною.

Для навчання нейронної мережі використано середню квадратичну помилку (*MSE*) як функцію витрат:

$$MSE = \text{Сума } [(\text{Прогноз} - \text{Фактичний})^2] \cdot (1 / \text{кількість\_спостережень})$$

Середньоквадратична помилка моделі є кількісним показником, який визначає кількість похибок шляхом квадратування дельти помилок результату нейромережі. Іде чисельна перевага на користь правильних результатів, та пригнічуються неправильні. Лінійна та логістична регреесія працюють за схожим прикладом.

Для подальшої оптимізації може бути використаний градієнт функції, який показує напрям оптимізації.

Градієнт функції обчислюється та представляється вектором який складається з похідних відносно усіх параметрів. Вектор дещо схожий на звичайні інструменти теорії лінійного програмування:

$$\text{Градієнт } C\,(B_0, B_1) = [dC / dB_0,\ dC / dB_1]$$

За допомогою такого градієнта можна повністю визначини бажаний напрям руху та можливі варіанти оптимізації цільової функції. Такий вектор також зазначає напрямок оптимізації та відносне числове значення у заданих одиницях:

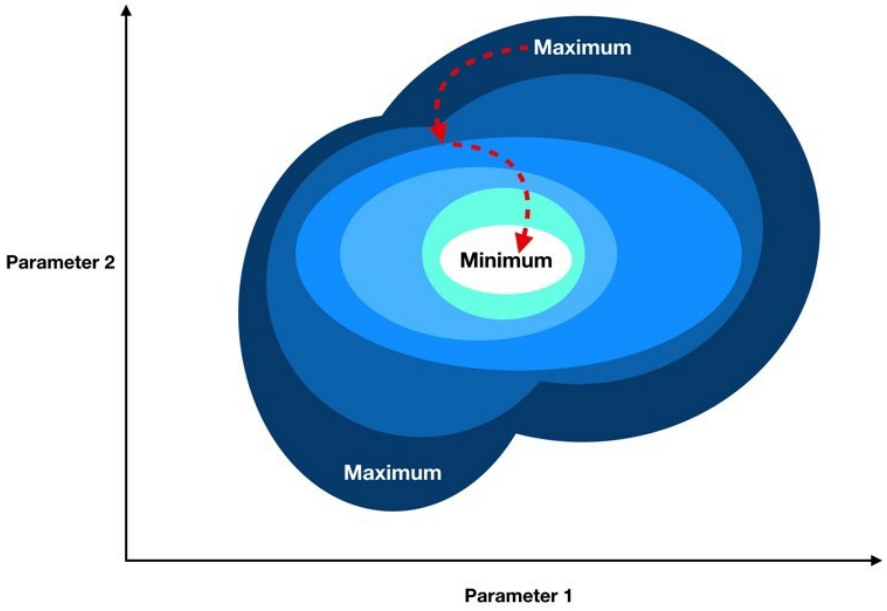


Рис. 2.7. Ілюстрація градієнтного спуску

З отриманим числовим значенням градієнта, необхідно змінити кожен параметр нейромережі на певну дельту — величину відносну до елемента градієнта у протилежному напрямку. Цей процес нагадує оптимізацію за допомогою частинних похідних функцій. Після проведення певних дій можна отримати задовільний результат, який буде відображати покращений показник на відмінну від попереднього отриманого. Впродовж ітерацій, показник буде тільки покращуватись.

Повторне обчислення градієнту, використовуючи нові налаштовані значення параметрів, і повторне проходження попередніх кроків буде проводитись, поки не досягнемо мінімуму.

2.3. Зворотне розмноження

Пряме поширення визначається процесом розрахунку значень у напрямі від початку до кінця. Зворотне поширення — визначається розрахунком повторних значень у іншому напрямку. У загальній конфігурації прийнято переміщувати помилку назад через модель.

Вхідні дані завжди модифікуютьс явагами, ухилами та функціями активації кожних нейронів. Вже за допомогою отриманого значення, відбувається розрахунок виходу нейрону.

Вихідна функція активації позначає останню функцію активації у цілому ланцюгу нейромережі.

Активації обчислюються на кожному кроці на кожному нейроні, для кожного прихованого шару, і врешті решт отримується необхідний для подальших дій результат.

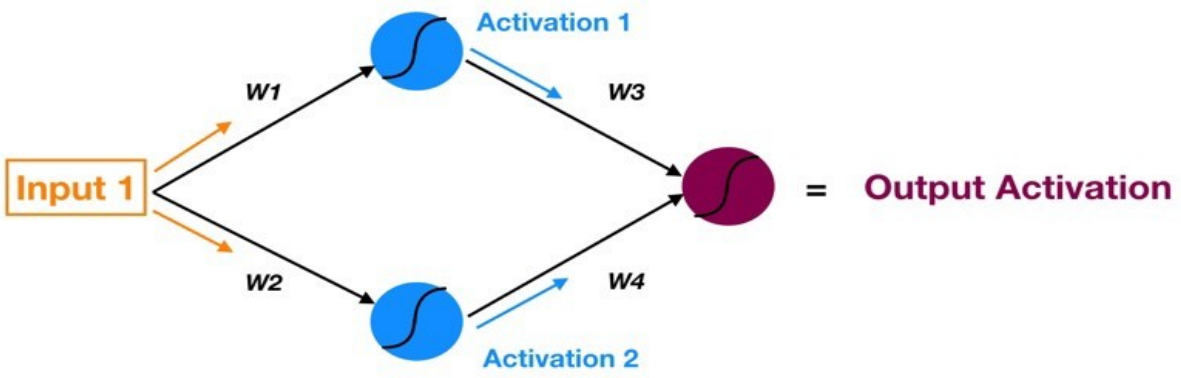


Рис. 2.8. Поширення вперед у нейронній мережі

Якщо слідувати червоним стрілкам (на малюнку нижче), помітитно, що зараз починаємо з виходу пурпурового нейрона. Це активація на виході, яку використовуємо для прогнозування, і остаточне джерело помилок у моделі. Потім переміщуємо цю помилку назад через модель за допомогою тих самих ваг і з'єднань, які використовуємо для прямого розповсюдження сигналу (тому замість Активації 1, тепер є *Error*1 – помилка, що приписується верхньому синьому нейрону).

Мета таких математичних перетворень полягає у розрахунку вихідного результату активації нейронів, поки не буде отриманий необхідний результат. Тепер можемо сформулювати мету зворотного розмноження подібним чином:

Похибки нейрону можна обчислити за допомогою власних або загальноприйнятих формул, показників ефективності нейромережі, відностих величин кількості правильних відповідей, тощо. Такі дії треба зробити на кожному кроці моделі.

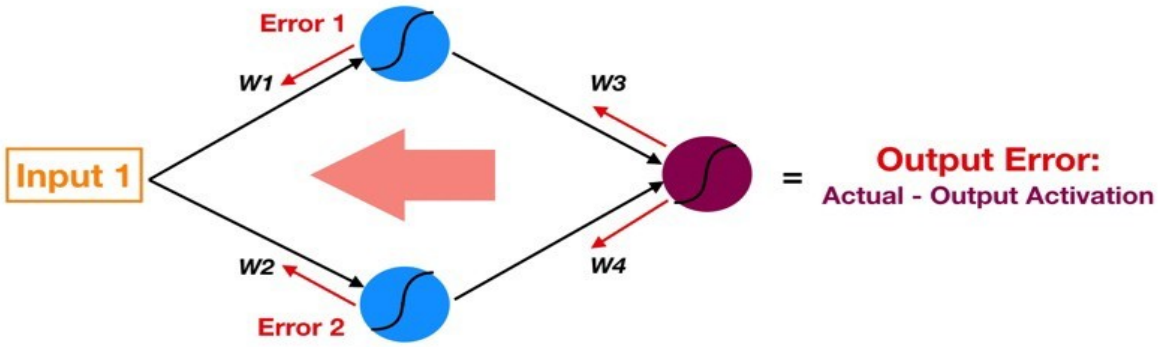


Рис. 2.9. Зворотне розмноження в нейронній мережі

Нейрони обмінюються інформацією між собою через їх зв’язки, які передають сигнали поміж ними. Ці зв’язки також мають ваги. Зміна цих ваг приведе до коригування роботи нейромережі та може змінити видані прогнози цієї моделі.

Величина похибки одного конкретного елемента мережі прямопропорційна впливу виходу обчислюванного елементу на активаційну функцію.

Таким чином, можна зробити висновок, що помилка кожного нейрона корелює з вхідними даними цього нейрона. Таке заключення виглядає досить логічним, якщо один елемент системи помиляється найбільше за усіх, тоді розрахунки цього елемента порушують роботу усієї системи у цілому та такий нейрон необїхідно виправити.

А часткові похідні визначають ступінь помилок та дають підказку як саме можна коригувати роботу нейронної мережі. Отже, в основному зворотне розповсюдження фактично розраховує похибку, що приписується кожному

елементу, таким чином можна виправити та розрахувати часткові похідні і градієнт для подальшої оптимізації.

2.4. Метод навчання «Гра за звинувачення»

Нейрони через зворотне розповсюдження діють за правилами гри у звинуваченні (рис. 2.10). Кожна помилка може бути розрахована та адресована на попередні активовані елементи системи, які винні у активації поточного нейрону. Таким чином можна відокремлити усі неправильні елементи системи та підкоригувати їх.

Найбільша похибка відокремлюється у тих елементів, які подали найбільш сильний сигнал до поточного нейрону та найчастіше активувався. Так можна побачити взаємозв’язок між елементами усієх нейромережевої прогнозувальної системи.

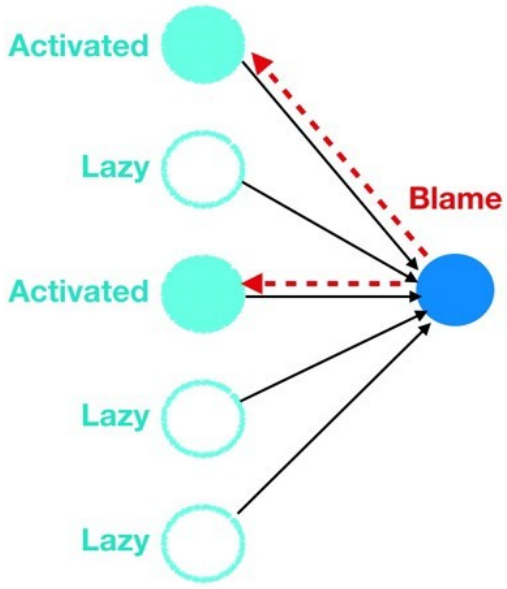


Рис. 2.10. Нейрони звинувачують найбільш активні нейрони вище по течії

Таким чином, робиться висновок, що найбільшу зміну ваг та параметрів відчувають саме активні нейрони, у той час як пасивні нейрони не реагують на рішення інших елементів системи. Хоча такий алгоритм і не здається оптимальним за його допомогою реалізовано багато вдалих рішень у

нейромережевих технологіях. Таким чином системі вдається досягти максимум ефективності.

За допомогою проведення таких дій можна шар за шаром виправити усі наявні помилки у нейромережі.

Помилка, пов’язана з певним нейроном напряму впливає на кількісне числове значення цього нейрону і повинна бути вирішена у той самий час як тільки це значення було отримано.

Існують модифікації методу, де пасивні нейрони також коригують свої ваги виправляючи вади своєї пассивності.

За такою схемою можна визначити окремі ознаки зображення, що описуються як відмінна риса або властивість зображення. Деякі такі характеристики є інтуітивно зрозумілими, коли інші можуть здаватися непримітними і можуть ігноруватися людським оком. До таких ознак відноситься яскравість, текстура, форма та контури об’єктів. Саме завдяки їх виокресленню відбувається класифікація зображення за певними характеристиками. Як приклад можна згадати гістограми розподілу яскравості.

2.5. Ознаки зображень для передачі у нейронну мережу

2.5.1. Ознаки яскравості зображень

Яскравість — одна із найбільш важливих ознак зображення. Ця характеристика представляється через спектральну інтенсивність випромінювання, координати кольору і т. д., які називатимуться ознаками яскравості. Вимірювання яскравості проводиться згідно з спеціальними формулами. Як приклад, середня яскравість околиці крапки $\{j, k\}$ даного зображення елементів визначається як:

$$Y(j,k) = \left(\frac{1}{(2W+1)^2} \right) \cdot \sum_{m=-W}^{W} \sum_{n=-W}^{W} Y(j+m, k+n) \qquad (2.1)$$

Ознаки яскравості можна визначити по різному. Можна використовувати чисельні розрахунки яскравості або значень кольору безпосередньо або перейти до інших ознак яскравості, виконавши різні види та різні способи числових перетворень.

Саме завдяки ознакам яскравості можна визначити основні ознаки об’єкту та зображення і робити більш детальне прогнозування щодо розташування цих елементів.

2.5.2. Просторово спектральні ознаки

Спектральне перетворення дає можливість отримати спектральні коефіцієнти які можуть знадобитись для визначення ваг базисних функцій отриманих згідно з цим перетворенням, при яких зважена сума таких функцій ідентична. Можна зробити висновок, що було отримане значення кореляції відповідних базисних функцій зображення. Зміна значення числових параметрів та коефіцієнтів дає чітко зрозуміти можливість висновку та виводу параметрів щодо зображення. Хоча на практиці такий процес і може вважатися складним, все ж таки при вдалому плануванні та коригуванні початкових параметрів можна досягти найкращого результату. Практична складність обумовлюється незвичайною формою та характеристиками яскравості об’єктів та можуть не представлятися простими базисними функціями. Хоча, існують випадки коли перетворення здійснюється дуже просто.

Лендеріс і Стенлі [13, 14] за допомогою двовимірного перетворення Фурье отримали основні ознаки зображення. Для цього вони використовували спеціальний пристрій. Оптична система вираховує ці значення з використанням наступних формул:

$$w \cdot x \equiv \sum_j w_j x_j \qquad (2.2)$$

де w_j – просторові частоти. Оптичний прочитуючий пристрій дає на виході функції:

$$M(w_1,w_2) = |F((w_1,w_2))|^2 \qquad (2.3)$$

значення яких порівняно з інтенсивностю спектру світла. Варто зауважити, що функції і $F(x, y)$ є парою, зв'язаною однозначним перетворенням, тоді як

функція M (w_1,w_2) неоднозначно пов'язана з $F(x, y)$. Наприклад, функція не змінюється, якщо початок координат зрушується або дорівнює 0. Для деяких застосувань така поведінка функції може показатися вартою використання.

Використання кута функції на площині частот дає просторово-частотну характеристику, незалежну від зрушення та шумів. Перевівши дані результати в полярні координати, з'являється результат у наступному вигляді:

$$N(p) = \int_0^{2\pi} M(\rho,\theta) d\theta \tag{2.4}$$

де θ – це змінна інваріанти щодо зміни масштабу:

$$P(\theta) = \int_0^{2\pi} M(\rho,\theta) \rho d\rho \tag{2.5}$$

При використанні зображення обмеженого доступу, поле перетворення Фур'є буде гаснути у певній множині. Якщо функцію конкатенувати з її вікном, можна пояснити то фур'є-спектр цього добутку рівний згортку з Фур'є-спектром функції вікна $|W(w_1,w_2)|$. Загасання через вичерпну границю вікна, слід враховувати при визначенні відносних значень коефіцієнтів Фур'е на різних просторово-часових частотах [6].

Виділення ознак зображення полягає у акцентувані уваги на областях конкретної форми. Принцип Фур'е для різних областей визначається таким чином:

1) Горизонтальна щілина:

$$S_1(\theta) = \int_{w_y(m)}^{w_y(m+1)} M(w_x, w_y) \rho dw_y \tag{2.6}$$

2) Вертикальна щілина:

$$S_2(\theta) = \int_{w_x(m)}^{w_x(m+1)} M(w_x, w_y) \rho dw_x \tag{2.7}$$

3) Кільце

$$S_3(m) = \int_{p(m)}^{p(m+1)} M(\rho,\theta) d\rho \tag{2.8}$$

4) Сектор

$$S_4(m) = \int_{\theta(m)}^{\theta(m+1)} M(\rho,\theta)d\theta \qquad (2.9)$$

Для комп'ютерного зображення, що описується *F (j, k)*, можна розглядати дискретно обчислювальний спектр Фур'є:

$$F(u,v) = \left(\frac{1}{N}\right) \cdot \sum_{m=-W}^{W} \sum_{n=-W}^{W} Y(j+m,k+n) \qquad (2.10)$$

В цьому випадку показники для вертикальної щілини, горизонтальної щілини можна визначити аналогічно виразам (2.6) – (2.9). Така ідея вже використовується також і у інших перетвореннях, зокрема, таких, як перетворення Адамара і перетворення Хаару [8]. Ознаки представлені у виді спектральних значень, вони були досліджені у теоретичних та практичних завданнях, в яких ці ознаки застосовувались як вхід для системи розпізнавання образів. Спектральні ознаки використовуються у класифікації різних галузей людської діяльності, зокрема у діагностиці хвороб по рентгенівських знімках [15, 16, 17].

2.5.3. Контурні ознаки

Різкі перепади забарвлення, координат яскравості або інших характеристик, що описують текстуру, є основними простими ознаками, оскільки вони зазначають контури зображень та об'єктів на них. Перепади значень яскравості у окремих місцях називаються локальні перерізи яскравості, або контурами яскравості (*luminance edge*). Протяжні розриви визначаються відрізками меж зображених на фото об'єктів. Розглянемо перепади яскравості у областях з однаковою яскравістю.

У зальному випадку перепад описується кутом нахилу та за допомогою координатою схилу. Перепад присутній коли його кут і висота перепаду більше ніж деякий заданий поріг. Для двох та багатовимірного перепаду важлива також розташування перепаду по відношенню до осі *x*. Ідеальний детектор перепадів при обробці частин зображення мусить акцентувати на наявність перепаду в конкретному пікселі, який знаходиться по центру схилу.

Властивий підхід до знаходження таких перепадів на монотонному зображенні описується у вигляді блок-схеми на рис. 2.1. Зображення, описане рядом чисел *F* (*j, K),* трансформується у лінійній обробці з тим, щоб визначити такі перепади. Як вихід, утворюється ряд інших чисел *G* (*j,* до), що іллюструє картинку з описаними змінами яркостей. Надалі виконується операція порівняння із зазначеними умовами і обраховується відносне положення об'єктів з перепадами[9]. Якщо

$$G(j,k) < T_L(j,k) \tag{2.11}$$

то має місце нижчий перепад, а при

$$G(j,k) = T_L(j,k) \tag{2.12}$$

вищий перепад. Величини *TL* (*j, k*$_A$) і *Ti* (*j, k*) є нижні і верхні порогові величини. Ці величини розглядаються зазвичай як змінні в площині зображення для нехтуванням впливу більш впливових змін яскравості на результати знаходження перепадів. Знаходження такого порогу є одним з ключових питань знаходження перепадів. При черезмірному рівні порогу деякі важливі елементи не можуть бути виявлені. Навпаки, занадно знижений рівень порогу буде наслідком того, що перепади можуть буди помилково визначені[11]. Для зображені перепадів на піддослідній частині можуть використовувати так званий контурний препарат — ряд чисел-елементів *E* (*j,k*). Наприклад, положення вищих перепадів описується білими крапками на сіруватому фоні. Точки вищих препаратів описуються білим кольором, нищих – чорним, а інші частини зображення – деяким середнім нейтральним кольором.

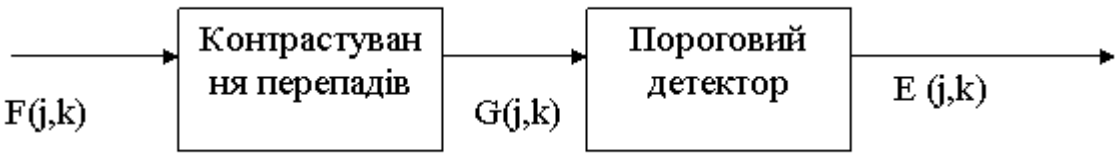


Рис. 2.11. Порогова система виявлення перепадів

Найбільш суттеве виявлення перепаду значиться у приблизному наближені частини реального зображення певним n-вимірним перепадом. Якщо приближення описується точно, то розуміється, що перепад існує і тому окреслюються нові параметри перепадів.

2.6. Прив'язка зображень

У безлічі застосунках та практичних розробках обробки зображень варто постійно тестувати та порівнювати n зображень одного і того ж об'єкту, які були зчитані за допомогою різних пристроїв, або n зображень об'єкту, отриманих з різних джерел та й у різних умовах. Для такого порівняння існує безліч методів, серед яких і автоматичні. Деякі просторові відмінності таким чином нехтують. До таких належать відмінності в обробці, зсуви, спотворення, а також геометричні перетворення і зміна контрасту картинки. Тонке налаштування датчиків зчитування інформації дозволяє униктути та недопустити неправильну інтерпретацію даних. На жаль, помилки все ж трапляються і треба виконувати виправлення. Для цього застосовують алгоритми виправлення [20]. У даній науковій праці проблема розв’язується шляхом поєднання зображень при присутності просторового зрушення. Отриимані відповіді застосовують для зменшення відмінних рис, викликаних трансформаціями і зміною масштабів, якщо збільшити масштам зображення або здійснити відносне перетворення. (Інформація може втрачатись під час переходу до іншого представлення, системи координат, тощо.)

Загальноприйняте рішення кореляційної прив'язки декількох величин лежить у тому, що визначається величина, що вимірює значення поміж існуючих значень, і знаходить локальний максимум такої функції [11, 12].

Розглянемо приклад цього методу у випадку нижче. Існують ряди чисел $F_1(j,k)$ і $F_2(j,k)$ які описуються n дискретними зображеннями, які потрібно віднести та вирахувати коефіцієнт приналежності один до одного.

$$R(m,n) = \frac{\sum_{j=1}^{M}\sum_{k=1}^{N}\left[F_1(j,k)F_2(j-m,k-n)\right]^2}{\sum_{j=1}^{M}\sum_{k=1}^{N}\left[F_1(j,k)F_2(j-m,k-n)\right]^2 \sum_{j=1}^{M}\sum_{k=1}^{N}\left[F(j,k)\right]^2} \qquad (2.21)$$

де (*j,k)*— порядкові номери об’єктів у вікні *W* розміром *JXK* штук, яке знаходитьс повністю у зоні *S* та має величину *MXN* об’єктів. Рис. 2.6 зображує співвідношення між частиною поля пошуку і вікном.

Функцію кореляції визначається та обчислюється для певних $(m - j + 1)$ $(n - k + 1)$ існуючих вікнон у певній частині пошуку для того, щоб розрахувати її значення й підрахувати оцінку такої похибки.

Такі прості методи та способи кореляції визначаються двома основними завданнями. Спочатку функція кореляції вираховує локальний максимум, який може бути важко виявити. Акцентуємо, що міра кореляції та структура порівнюваних зображень відрізняється. Також, можливі недоліки, биті пікселі, на зображенні може помилково видати максимум кореляції. З цим борються шляхом подолання, покращивши кореляцію таким чином, щоб похибка була мінімальна $F_1(j, k)$ і $F_2(j, k)$.

Краща міра кореляції задається:

$$R(m,n) = \frac{\sum_{j=1}^{M}\sum_{k=1}^{N}\left[G_1(j,k)G_2(j-m,k-n)\right]^2}{\sum_{j=1}^{M}\sum_{k=1}^{N}\left[G_1(j,k)G_2(j-m,k-n)\right]^{1/2}\sum_{j=1}^{M}\sum_{k=1}^{N}\left[G(j,k)\right]^{1/2}} \qquad (2.22)$$

де ряди чисел $G_i(j, k)$ виходять рядами, що характеризують зображення, з методами $D_i(j, k)$, що є характеристиками фільтрів:

$$G_i(j, k) = F_i(j, k) \cdot D_i(j, k) \qquad (2.23)$$

Такі існуючі характеристики розраховуються так, щоб мінімізувати можливу кореляцію коли можна виконати такі і подальші дії найкращим та найліпшим чином.

f_1 — ряд чисел, функція генератор отримана у результаті роботи з такою функцією $F_1(j, k)$, відповідного вікна, а ряд чисел $f_2(m, n)$ який складається з певних фрагментів $F_2(j, k)$ при визначеному зміщенні (m, n).

Така кількість різних рядів $f_2(m, n)$ складає $M*N$. Елементи векторів f_1 і f_2 (m, n) зазвичай корелюють між собою.

Як правило, до методу вигодженої фільтрації таких полів спочатку іде обробка, яка містить у собі відбілення, тобто, у рядах таких векторів на виконуються операції додаванян фільтрів H_1 і H_2:

$$g_1=[H_1]^{-1}f_1 \qquad (2.24)$$

Як і перцептрон, сигмоподібний нейрон має входи x_1, x_2, x_3. Але на відмінну від того, щоб визначати або 0 або 1, такі рішення можуть визначати лише певні значення у діапазоні від 0 і 1. Як ось, 0,638 – є притаманним входом для такого лінійного нейрона. Зазначимо, що як і перцептрон, лінійний нейрон має параметри для кожного значення $w_1, w_2, \cdots$ і загальне зміщення b. На виході ситуація дещо інша. Натомість це:

$$\sigma(w \cdot x + b),$$

де σ – називається сигмоподібною функцією.

До речі, σ такая функція яка використовує логістичну регрессію, а такий новий сет нейронів прийнято вважати логістичними. Треба акцентувати на цій термінології, оскільки такі терміни дуже поширені та зустічаються у всіх наукових працях з нейромереж та розпізнавання образів. Проте будемо брати до уваги такі термінології:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Така сигмоїдальна функція описує вихід сигмоподібного нейрона з входами x_1, x_2, вагими w_1, w_2 та зміщенням b є:

$$\sigma(z) = \frac{1}{1+e^{-\sum_j w_j x_j - b}}.$$

Здавалося б, сигмоподібні нейрони представляються дуже різними від та зовсім не схожими на перцептрони. Формула сигмоїдної функції не ясна, неконкретна, якщо судити з першого погляду. Насправді між такими перцептронами присутньо багато спільних рис та математична форма сигмоподібної функції розуміється більше технічною деталлю, аніж справжньою проблемою для сприйняття.

Припустимо, щоб зрозуміти схожість із моделлю перцептрону $z \equiv w \cdot x + b$ – певне додатне число. Тоді $e^{-z} \approx 0$ і так $\sigma(z) \approx 1$.

Інакше кажучи, коли $z = w \cdot x + b$ має найбільше значення і більше за 0, вихід із сигмоподібного нейрона досягає 1, як і для перцептрона.

Дивлячись інакше, що $z = w \cdot x + b$ набуває дуже малого значення. Тоді $e^{-z} \rightarrow \infty$, і $\sigma(z) \approx 0$.

Якщо $z = w \cdot x + b$ дуже малого значення, така поведінка нейрона також схожа більш до перцептрону. Якщо $w \cdot x + b$ має помірні розміри, що значно обмежуються порівняно з моделлю перцептрону.

Коли σ здається була б помірною функцією, то такий подібний нейрон був би перцептроном, якщо вихід такого результату є або 1 або 0 дивлячись, чи $w \cdot x + b$ було позитивним чи негативним.

Якщо $w \cdot x + b = 0$ перцептрон визначає 0, тоді як такий спосіб кроку видає 1. Отже, можна зробити висновок, *що* варто було б поміняти таку функцію кроку у цій та у іншій точці. Використовуючи σ-функції можна використати такий перцептрон. Такі значення σ функції є значущими та результатом, а не маленькою формальністю.

Розрахунки проводяться відповідно та послідовно. Якщо таке значення та такі відмінності сильно більше відрізняються від певного порогу, то J x N об’єктів у цих вимірах будуть підраховані таким чином, що розрахунки для даного виміру не обмежувалися, і перевірялися також інші виміри. Таке значення може трохи зростати, і число таких точок, розрахованих на попередньому кроці, коли, нарешті, таке значення буде вирахувано, записується і відзначається як характеристика випробуваного виміру. Дослідження усіх вимірів з найвищою оцінкою вважається зробленим якщо розраховується в коректній позиції. Максимально можливе значення дорівнює такій кількості крапок у вікні розміром *JXK*. Слід зазначити, що у разі виміру, яким зображують великі неточності, потрібне достатьо незначне число розрахунків.

Для оптимізації швидкості зсходу алгоритму і покращення надійності було прийнять декілька додаткових налаштувань алгоритму для випробувань [18]. Найпросішою служить гібридна система, де точки ігноруються з значними

похибками використовується алгоритм послідовних відсікань з подальшим розрахунками міри таких кореляції для точок, що залишилися та потребують обчислень. Така система поєднує переваги усіх методів обчислень, робить кращими показники кореляції і надає значну швидкость методу послідовних випробувань.

Вираз все ще може здаватися занадто складним та незрозумілий звичайному науковцю, але Δ є лише лінійною функцією змін Δw_j у вагах і Δb зміщенні. Така форма покращує варіанти незалежних змін значень та відхилів для обчислення будь-якої необхідної невеликої зміни у усій потужності. Так, хоча і такі нейрони мають приблизно таку ж орієновну поведінку, як і перцептрони, вони значно обширно використовуються для з'ясування того, як зміна ваги та зміщення змінить результат.

Як зазначалося, такий з краю шар у цій системі має назву вхідним шаром, а нейрони поміж цього шару – вхідними нейронами. Сусідні праві або вихідний шар має у собі вихідні нейрони або, як і у іншому варіанті, один вихідний нейрон. Проміжні шари позначаються прихованими шарами, так як нейрони в цьому випадку не є ні вхідними, ні вихідними.

Зображення вхідного та вихідного шарів у системі є спрощеним. Як ось, наприклад, припустимо, що нібито спробуємо розрахувати, чи дійсно на такому зображенні «5» чи ні. Загальноприйнятим способом системи є представлення інтенсивності пікселів картинки у вхідні нейрони. Зображення розміром 64 від 128 зображень у градаціях сірого, тоді б мали 8192 = 128 × 64 вхідні нейрони, робота яких зосереджується на відрізках відповідно 0 і 1.

Хоча власне така робота вхідного та вихідного слоїв нейронної мережі часто є загальноприйнятою, у дизайні таких частин може бути ціла наука. Особливо неможливо зробити підсумки роботи проектування скритих шарів за такими правилами . Навпаки, дослідники мереж намагаються втілити у реальність безліч евристик проектування шарів, які допомагають зробити поведінку, яку треба зробити із своїх мереж.

Проте лише інші моделі штучних нейронних мереж, і лише у певних можливі такі звороти зворотного зв'язку. Ці моделі описуються як рекурентні

нейронні мережі. Суть цих моделей у наявності нейронів спрацьовуючих з певним обмеженим та лімітованим періодом часу, перш ніж дійсно активуватись. Цей потік може активувати інші нейрони, які можуть згаснути трохи пізніше, також протягом лімітованого часу.

2.7. Нейромережеві алгоритми

Нейромережеві методи – комплекси алгоритміів, що використовуються на застосуванні певних типів нейронних мереж (НМ). Такі напрями використання різних НМ для таких образів і зображень:

– використання для таких “витягань” характеристик та параметрів або ознак образів;

– визначення до класу самих образів або вже таких з них характеристик (у першому випадку такі дії відбуваються непомітно та відбувається неявно усередині мережі);

– рішення задач оптимізації.

Архітектура таких НМ має певну збіжність з природними та штучними нейронними мережами. НМ, які орієнтовані для вирішення таких завдань, можуть істотно бачитися як алгоритмами функціонування, але їх головні властивості наступні.

НМ складається з частин, так називаємих неіснуючих нейронами, які самі по собі виглядають як прості та лише поєднані іншими нейронами. Кожна така частина перетворить послідовність сигналів поступаючих до нього на початок у вихідний сигнал у кінець. Такі зв'язки між такими нейронами, що можуть визначатися вагами, грають ключову роль [12]. Одна з вдосконалень НМ (а так само недоопрацювання при такій їх не послідовній системі) це те, що всі такі об’єкти можуть лише працювати паралельно та узгоджено, таким чином істотно покращуючи рішення поставлених задач, особливо в аналізі зображень. Крім того, що НМ дозволяють таким чином власе вирішувати безліч проблмм, вони представляють сильні, гнучкі і такі механізми навчання, що є їх такою перевагою перед такими методами (імовірнісні методи, лінійні перепони, дерева рішень і

т.п.). Навчання позбавляє від необхідності вибирати ключові ознаки, їх значущість і відносини між ознаками. Але проте вибір початкового представлення вхідних даних (вектор в *n*-мірному просторі, частотні характеристики, вейвлети і т.п.), істотно впливає на якість рішення і є окремою темою. НМ володіють такою здатністю для обгрунтування (краще ніж за такі собі вирішальні дерева). Також можливо якісно пропагандувати досвід, такий що здобутий на кінцевому навчальному наборі, на всю низку образів.

Для розпізнавання людини по зображенню обличчя рекомендовано використовувати багатошарові нейронні мережі (БНМ).

Архітектура нейронної мережі (БНМ) містить у собі сполучені частинки, у той час як нейрон кожного шару пов'язаний з такими всіма іншими нейронами такого попереднього шару, а виходами – наступного. НМ з декількома шарами для вирішення може з певною якістю визначити будь-яку задану та відносно просту багатовимірну функцію. НМ з таким одним вирішальним шаром може лише формувати такі лінійні розділяючі поверхні, що розділяють можливі рішення на 2 категорії, таким чином такі мережі не можуть розв’язати задачу такого типу що або "включає або". НМ з такою нелінійною функцією активації і nма вирішальними шарами може формувати певні навіть випуклі області у мега просторі рішень, а з такими вирішальними шарами – області будь-якого об’єму , у тому числі і складності. При цьому БНМ не лише не втрачає своєї здатності розв’язувати завдання. НМ навчаються за допомогою алгоритму зворотного розповсюдження помилки, зокрема градієнтного спуску, що є частиною оптимізаційних методів у просторі вагів з методом зменшення фактичної похибки мережі. При цьому такі можливі та визначені похибки передаються у зворотному напрямку від входів до виходів, крізь параметри, що з’єднують нейрони [5].

Попри таке застосування одношарової НМ з асоціативною пам'яттю існує безліч в навчанні мережі проблем як саме відновлювати зображення, що слугуть як вхід до системи. Віддаючи на вхід таке зображення і розраховуючи таку якість вихідного зображення можна зрозуміти, як саме мережа розпізнала чи не розпізнала зображення на початку. З властивостей цього методу можна зазначити

саме в тому, що мережа може відокремити спотворені і “биті“ зображення, але для більш практичних задач він не задовільний [20].

Якщо ця позначка нижча за певне значення, то треба вважати, що такий образ не належить ні до одного з можливих класів. Процес такого навчання лише не може зосередитись на відповідності об’єктів, що таким чином задаються на вхід, з відношенням до певного класу. Це називається таким навчанням з вчителем. У лише застосуванні до розпізнавання маски по зображенню особи у масці, такий підхід чудовий для тільки завдань аутентифікації доступу невеликої кількості осіб. Такий підхід задовольняє безпосереднє відношення мережею таких і тих самих об’єктів, але із таким числом варіантів прогнозований час навчання збільшується за правилом експоненти. Тому під час виконання таких завдань, як пошук людини у базі даних, наприкад у прикордонній службі, вимагає такого витягу компактного набору власних параметрів, на базі яких можна здійснити пошук. Лиш напрям з використанням можливих характеристик всього зображення, описаний раніше в [7]. Для цього застосовувалася одношарова НМ, побудована на нейронах. Розмічене 100% можливе знаходження на базі даних університету, але при цьому здійснювалося класифікація серед картинок, яким мережа була натренована. Застосовується БНМ для відношення зображень осіб на базі таких можливих параметрів, як відстані між деякими звичайними частинами обличчя (ніс, рот, очі) [13]. Таким чином на вхід НС лише задавалися ці відстані. Буди використані так само інші методи – в першому на вхід НМ подавалися результати такої роботи прихованою ланцюговою моделлю, а в другому – висновок роботи НМ віддавався на вхід іншої ланцюгової моделі. У такому випадку недоліків не спостерігалося, що говорить про те, що розрахунки висновків НМ достатній.

Таке застосування НМ для задач пошуку зображень, коли на початку мережі поступають певні відомості декомпозиції картинки по методу головних компонент. У звичайній БНМ міжшарові нейронні з'єднання повні та обширні , і картинка представлена саме як одновимірний ряд чисел, хоча воно і неповне. Архітектура згорткової НМ вирішує усунення цих недоліків. У ній згадувалися

локальні поля реагування для забезпечння такої локальної двовимірної зв'язністі нейронів. Такі доступні ваги забезпечують знаходження деяких рис в будь-якому місці зображення і їх можлива реаліхація з просторовими сетами (*spatial subsampling*).

Згортальна НМ (ЗНМ) лише може надати постійну стійкість до змін масштабу, зсувів, поворотів. Сама власне ЗНМ складається з багатьох шарів, де шар має декілька площин, таким чином що нейрони таких шарів пов'язані тільки з певною кількістю нейронів такого шару з ближніх локальних областей (як в зоровій корі людини)[15]. Такі ваги в кожній просторі площини лише можуть бути згортальними шари. За згортальним шаром іде наступний слой, що може таким чином збільшити його шляхом усереднення. Процес повторюється знову і згортальний шар далі зменшує масштаби мережі. Таким чином, досягається циклічна зменшуюча організація. Пізніші шари розраховують більш загальні параметри, менше залежні від кількості таких спотворень зображення. ЗНМ коригується методом зворотного розповсюдження помилки.

Порівняння БНМ і ЗНМ може зобразити такі істотні переваги як за часом виконання , так і по якості класифікації. Властивістю такої можливої ЗНМ є і те, що параметри, що визначаються на виходах поверхневих шарів, лише мають бути такими для класифікації по способу найближчого сусіда (наприклад, розраховуючи манхетанську відстань), причому ЗНМ може і не дуже якісно витягувати такі параметри для об’єктів, які не були присутні в навчальному наборі. Для ЗНМ визначається та акцентується швидкість навчання і роботи. Тестуванні ЗНМ на базі даних, що має у собі картинки осіб з можливими змінами освітлення, розміру, просторових ознак, положення та вигляду, показало достатню 98% точність класифікації. Варто зазначити для відомих осіб, пред'являлися варіанти їх зображень, відсутніх в навчальному наборі. Такий результат робить цю роботу можливою для подальших аналізів в області розпізнавання картинок просторових сутностей.

БНМ застосовуються для знаходження картинок певного типу. Варто зауважити, що будь-яка така можлива БНМ в певній мірі може відрізнити

належність певних образів до "своїх" класів, також можливо заздалегідь навчити якісному детектуванню таких класів. В цьому варіанті вихідними ознаками будуть класи які мають і не належать до певного класу ознак[18]. Використання неромережевого детектору для найбільш вдалого знаходження зображення особи у вхідній картинці. Зображення оброблювалося вікном 10x10 пікселів, яке використовувалося на вхід мережі, та під час вирішального етапу чи належить дана ділянка до типу осіб. Навчання виконувалось як з таким із позитивних варіантів (різних зображень осіб), так і можливими (зображень, що не є особами). Для підвищення такої детектування застосовувалися колектив НМ, навчених з можливими попередньозаданими вагами, унаслідок чого НМ можливо лише робити висновок інашке, а остаточне рішення розраховувалося самме голосуванням усіх нейронів.

НМ використовується так само для таких ключових характеристик картинок, які потім можуть бути розглянені для подальшого пошуку. Тут наведений спосіб нейромережевої аргументаціїї методу загального аналізу головних компонент. Суть методу аналізу головних компонент полягає в підрахуванні максимально можливих коефіцієнтів, що збігаються з вхідними образами. Ці коефіцієнти згадуються як головними компонентами і розраховуються для статистичного стиснення картинок, в якому невелике число коефіцієнтів повідомляється для відображення всіх даних. НМ з одним можливим шаром з N нейронів (кількість шарів значна менше ніж розмір зображення), що може та навчена по методу такого розповсюдження помилки прирівнювати на виході картинки, подане на вхід, робить на виході висновок щодо нейронів коефіцієнти перших N головних компонент, які і моделюються для порівняння. Зазвичай розраховується від 10 до 200 головних компонент. Із збільшенням номера її результат сильно знижується, і таким чином компоненти з надлишковими номерами не має сенсу. При використанні нелінійних функцій нейронних шарів можлива нелінійна інтерполяція на можливі компоненти. Нелінійність може визначити точніше відобразити варіації вхідних даних. Застосовуючи аналіз таких компонент до інтерполяції зображень осіб, отримаємо впливові компоненти, звані власними особами, яким так само

визначають головний результат, – існують компоненти, які містять у собі всі характеристики та відображають такі ознаки як овочі, емоції. При відновленні компоненти мають характеристики, схожий на об’єкт, причому перші схожі найбільш загальну форму особи, останні – різні дрібні властивості між особами. Такий метод добре визначений для пошуку подібних зображень у базах даних google. Можлива так само подальше зменшення розмірності усіх можливих головних компонент за допомогою НМ. Таким чином вигляд реконструкції вхідного зображення позначається дуже точно і також визначається клас зображення.

2.8. Висновки до розділу 2

Розглянуто методи та принципи роботи нейронних мереж для визначення характеристик зображення.

Просте таке використання НМ у побутовому значені (званою асоціативною пам'яттю) визначається у навчанні мережі та її подальшої здатності відновлювати зображення. На вході присутнє тестове зображення і розраховуючи якість такого вихідного зображення, можна зробити висновок, наскільки така мережа може реконструювати вхідне зображення. Позитивні можливості цього методу, їх суть в тому, що мережа розпізнає нечітуі зображення, але для таких більш важливих цілей він незадовільний. Саме тому варто зосередитись на багатошаровій ШНМ.

РОЗДІЛ 3
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ВИЗНАЧЕННЯ МЕДИЧНОЇ МАСКИ НА ОБЛИЧЧІ

3.1. Архітектура системи

Програмне забезпечення визначення медичної маски на обличчі реалізує розпізнавання обличь на статичних зображеннях та озволяє іентифікувати їх за властивістю наявності медичної маски.

Бачучи що систему можна визначити як наступною такою діаграмою прецедентів (рис. 3.1). Дана діаграма зображає відношення між акторами та прецедентами в системі.

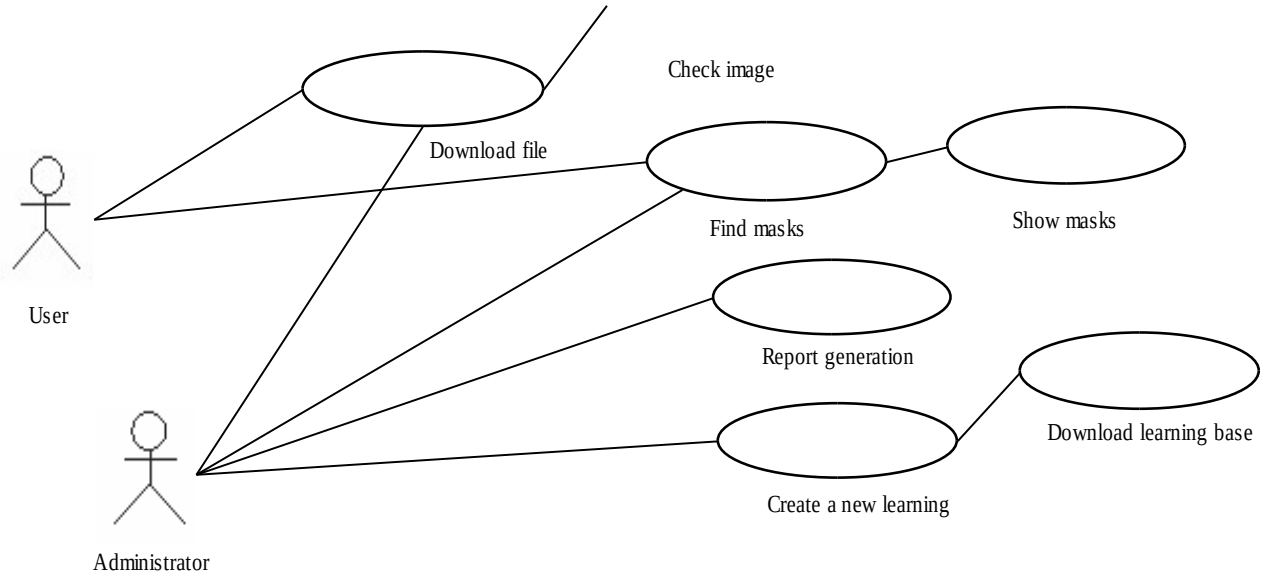


Рис. 3.1. Діаграма прецедентів використання системи

В системі передбачено 2 групи акторів, кожній з яких надаються різні рівні доступу:

1) *administrator* – роль, яка дозволяє не тільки в повній мірі використовувати систему, а також може запускати процес перенавчання;

2) *user* – роль, яка надається всім користувачам системи без реєстрації. Дана роль має доступ до о завантаження і аналізу зображень.

Для такої реалізації системи може визначити послідовність дій, яка таким чином для кожної можливості. Так можлива функція роботи з можливими даними у системі може бути зазначена наступними діями (рис. 3.2).

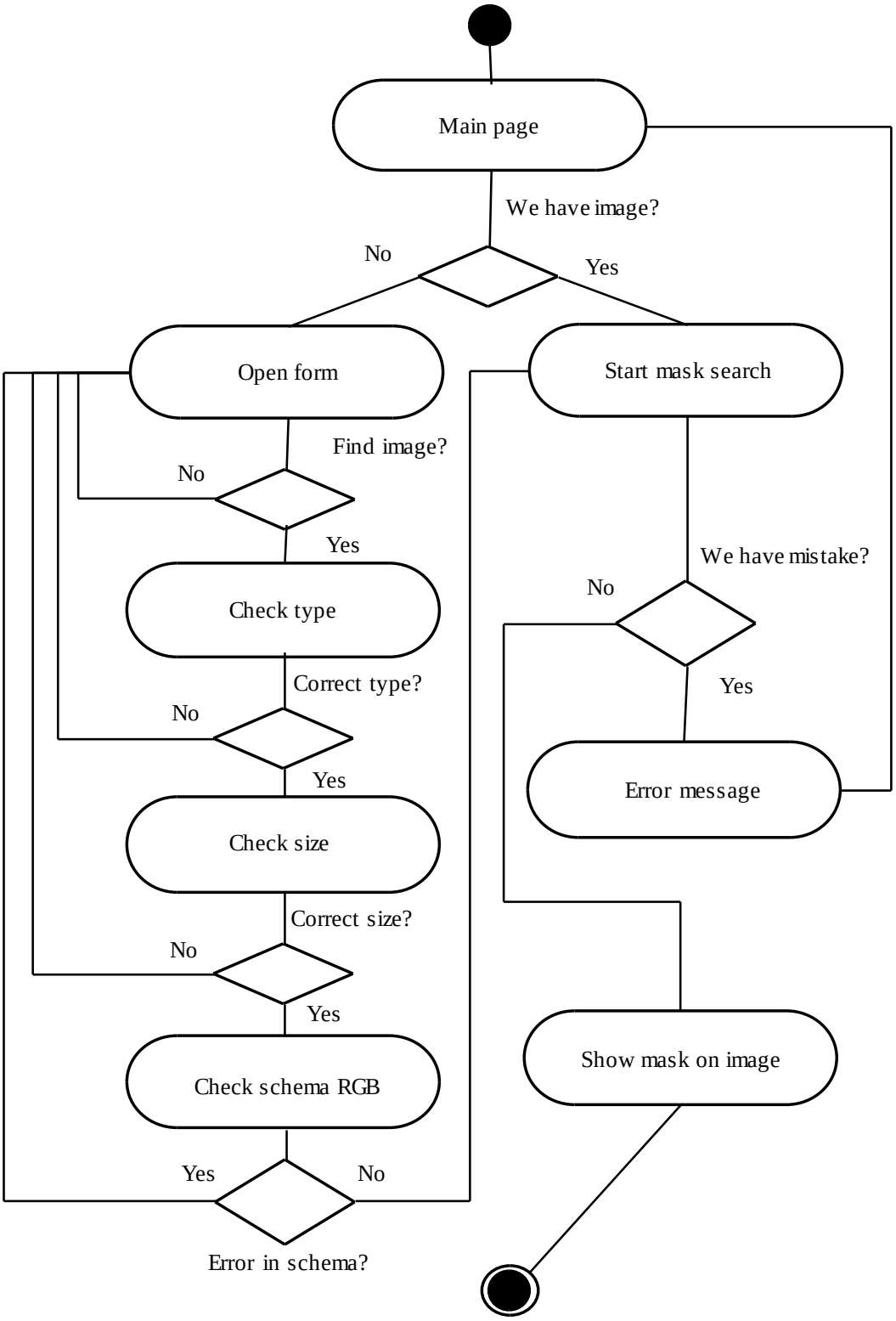


Рис. 3.2. Діаграма послідовності дій для запису даних в системі

Ціла структура взаємозв’язків між можливими класами таких можна представити вважати классами (рис. 3.3), яка описує взаємозв’язки між всіма класами системи..

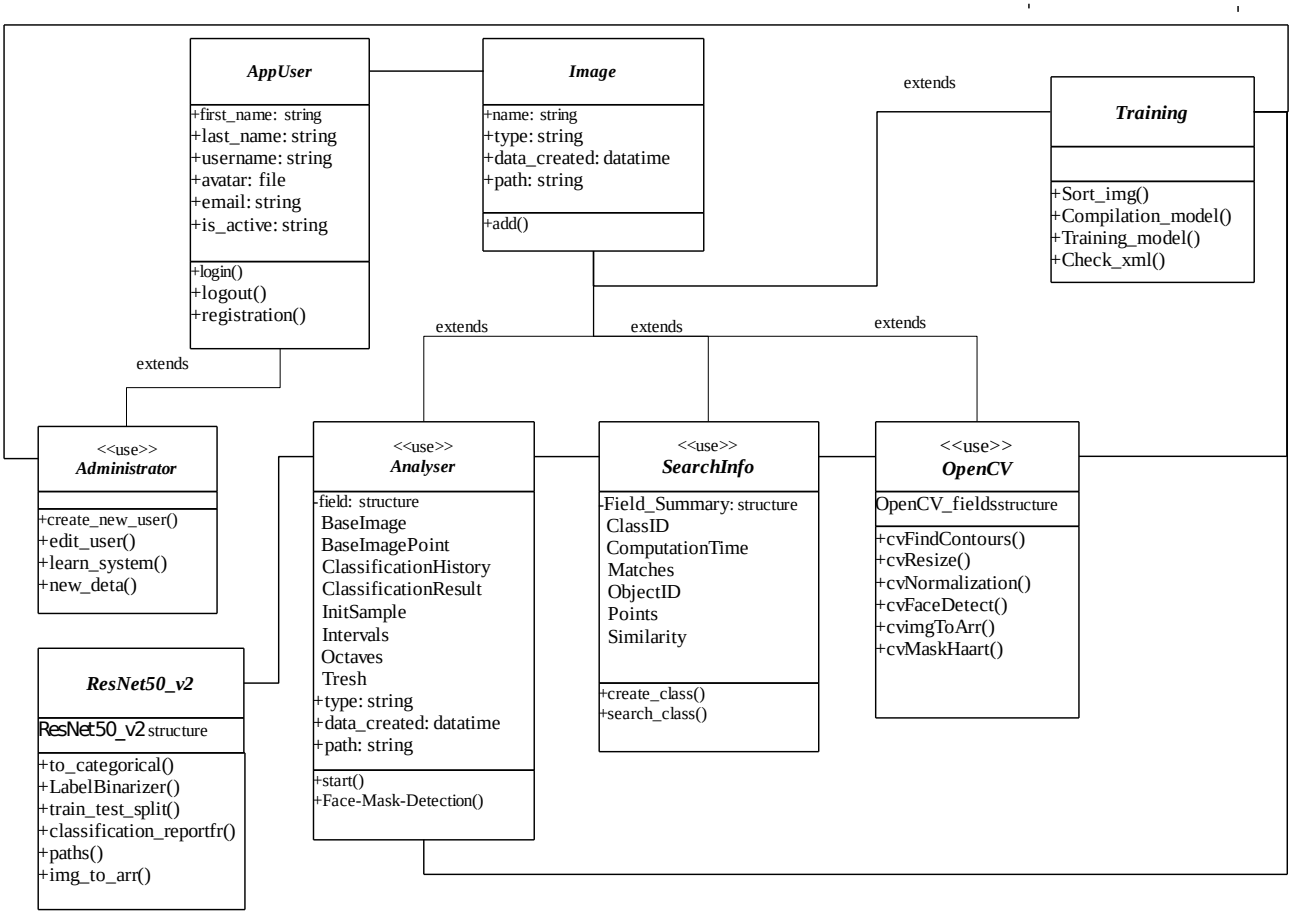


Рис. 3.3. Діаграма класів системи обліку стану здоров’я

При такій діаграми класів можлива покращена роботи системи і таким можливим до її реалізації.

На цьому кроці відокремлено основні важливі системи, які будуються з можливих проанілозованих можливостей використання системи.

Основними важливими системи є:

– *AppUser* – сутність користувача;

– *Administrator* – розширення для можливого визначення прав адміністратора;

– *Image* – сутність зображення;

– *Analyser* – аналізу зображень;

– *SearchInfo* – модуль пошуку класів по базі;

– *Training* – модуль тренування;

– *OpenCV, ResNet*50_*v*2 – інші бібліотеки.

3.2. Описання алгоритмів реалізації окремих задач у програмному забезпеченні розпізнавання маски на обличчі

3.2.1. Формування наборів даних

Для початку, щоб зробити детектор маски, потрібні відповідні дані. Крім того, через природу зовнішньої бібліотеки потрібні анотовані дані з прямокутниками, що обмежують. Один із варіантів — створити власний набір даних, або збираючи зображення з Інтернету, або фотографуючи друзів, знайомих та анотуючи фотографії вручну за допомогою певних програм, таких як *LabelImg*. Однак обидва варіанти трудомісткі, особливо останній. Є інший варіант, найбільш життєздатний для реалізації мети - працювати з публічним набором даних.

Базовий набір даних складається з 4100 зображень, які можна розділити на два класи: 2165 зображень з маскою та 1935 зображень без маски (рис. 3.4). Використані зображення були отримані з наступних відкритих джерел: *Bing Search API, Kaggle, RMFD*.

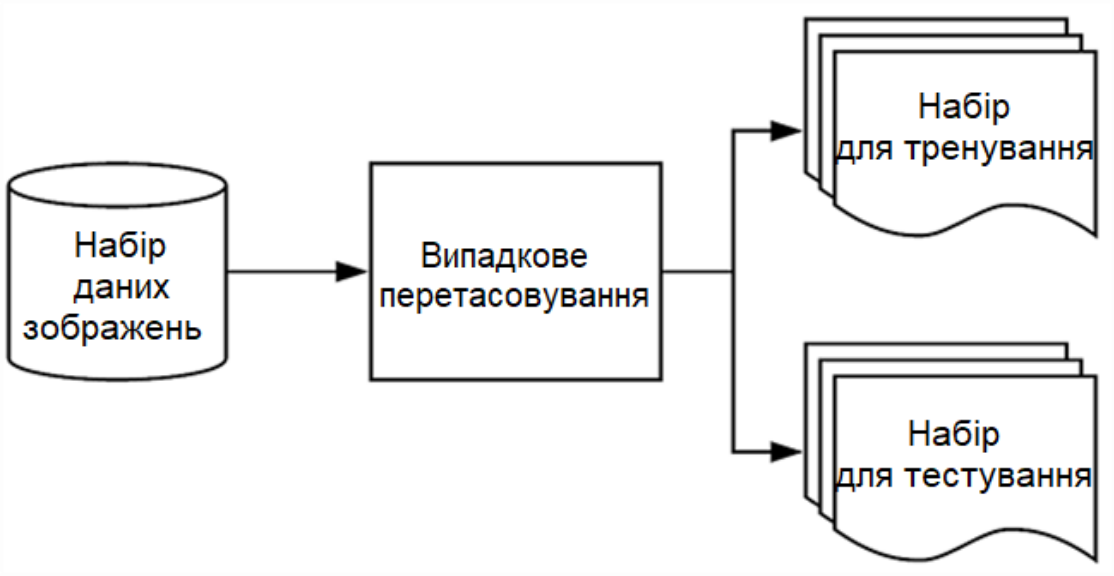


Рис. 3.4. Схема формування наборів даних для тренування та тестування

Після завантаження набору даних, щоб тренувати модель, потрібно перетворити файли *.xml* на *.txt*. Приклад:

<annotation>
<folder>images</folder>
<filenome>mymymymy.png</filename>
<size>
<wiidth>1024</width>
<heiight>768</height>
<depth>4</depth>
</zize>
<segmented>0</segmented>
<0bject>
<name>without_mask</name>
<pose>Un_specified</pose>
<truncoted>0</truncated>
<ocduded>0</occluded>
<difficult>0</difficult>
<bndbox>
<xnnin>77</xmin>
<ymin>103</ymin>
<xmax>109</xmax>
<ymax>142</ymax>
</bnddox>
</object>
<object>
<name>with_mask</name>
<pose>Unspecified</pose>
<truncated>0</truncated>
<occluded>0</occluded>
<difficult>0</difficult>
<bndbox>

<*xmin*>185</*xmin*>

<*ymin*>100</*ymin*>

<*xmox*>226</*xmax*>

<*ymox*>144</*ymax*>

</*bndbox*>

</*object*>

<*object*>

<ma*me*>*without_mask*</*name*>

<*pose*>*Unspecified*</*pose*>

<*truncated*>0</*truncated*>

<*occluded*>0</*occluded*>

<*ditticult*>0</*difficult*>

<*bndbox*>

<*xmin*>325</*xmin*>

<*ymiin*>90</*ymin*>

<*xmax*>360</*xmax*>

<*ymax*>141</*ymax*>

</*bndbox*>

</*odject*>

</*annotation*>

Ця анотація зображення, що містить лише 3 обмежувальні прямокутники, це видно за кількістю <*object*> ... </*object*> в *XML*.

Щоб створити відповідний текстовий файл, потрібно 5 типів даних кожного *XML*-файлу. Для кожного <*object*> ... </*object*> в *XML*-файлі необхідно використати клас з бібліотеки *OpenCV* і обробити поле <*name*>...</*name*> і координати прямокутника, що обмежує, – 4 атрибути в <*bndbox*>...</*bndbox*> . Відповідний формат виглядає так:

<*class_name*> <*x_center*> <*y_center*> <*width*> <*height*>

Наприклад, у *Image*1.*jpg* має бути відповідний *Image*1.*txt*, ось такий:

1 0.18224475 0.336989693977881 0.05989898 0.10108171617485569

0 0.4013669575 0.3333333333333333 0.080078125 0.12021857923497267

1 0.6689465525 0.3176737704927033 0.065559375 0.13933336229507996

Це точне перетворення вищезазначеного файлу *.xml* у відповідний текст.

Перш ніж розпочати навчання моделі, необхідно бути абсолютно впевненим, що перетворення було правильним. Щоб гарантувати відповідність, я було написано скрипт (*check_xml.py*), який бере зображення та відповідну йому текстову анотацію із заданої папки і відображає взяте зображення з прямокутниками, що обмежують. Даний скрипт використовувався для навчання бази (рис. 3.5).

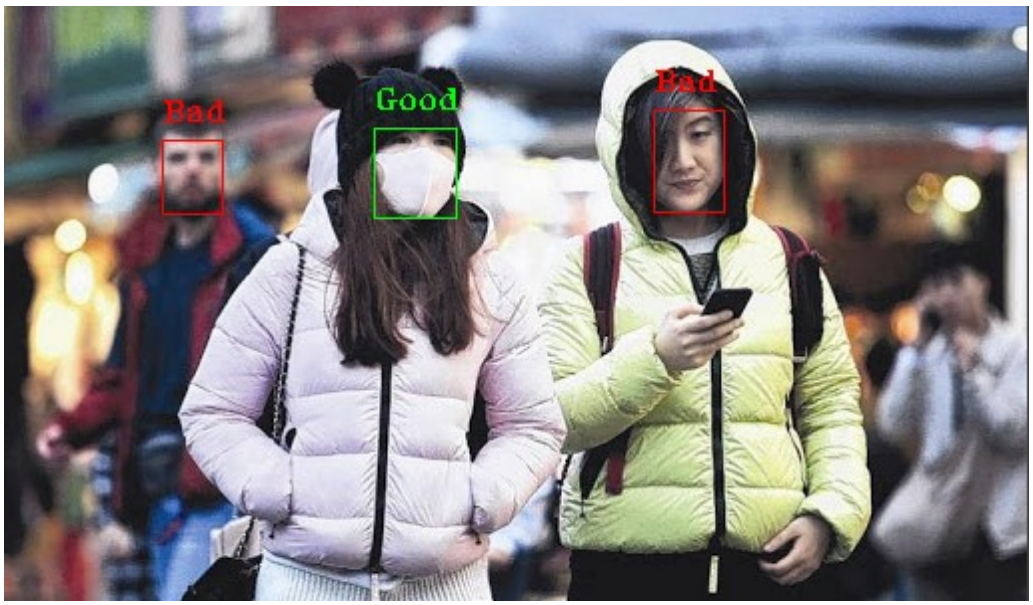


Рис. 3.5. Результат роботи скрипта *check_xml.py*

Щоб навчити модель і перевірити її на етапі навчання, дані було розділено на два набори – набір навчання та набір тестування. Пропорція склала 90 та 10 % відповідно.

У нашому випадку вихідний набір даних має 3 категорії: *with_mask, without_mask* та *mask_weared_incorrect* [з маскою, без маски, маска одягнена неправильно].

Щоб трохи спростити рішення було об'єднано дві останні категорії в одну. Отже, є дві категорії, *Good* та *Bad*, на підставі того, чи правильно хтось носить свою маску:

– *Good*;

– *Bad*.

face_mask.data: створіть файл *_.data*, який містить релевантну інформацію про класи, з ним працюватиме програма:

classes=2

tmin=data/train.txt

volid=data/test.txt

names=data/face_mask.names

badup= backup/

Навчання починалось з роздільної здатності 416*x*416 і навчено модель за 4000 ітерацій, але щоб досягти більшої точності, було збільшено дозвіл і продовжено навчання ще на 3000 ітерацій.

Було прописано прапорець *-map*, щоб у консоль виводилися важливі показники, такі як *average Loss*, *Precision*, *Recall*, *AveragePrecision* (*AP*), *meanAveragePrecsion* (*mAP*), тощо (рис. 3.6).

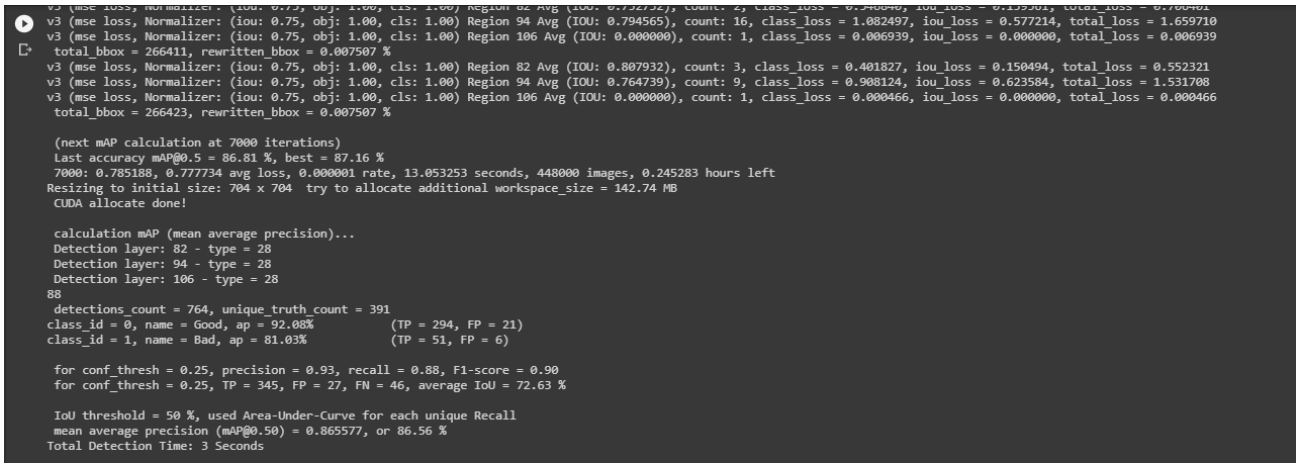


Рис. 3.6. Логування процесу обробки даних

За практико використання [6, 7, 12] індикатор *mAP* в консолі вважається кращою метрикою, ніж *Loss*, тому модель навчалась, доки *mAP* зростав.

Залежно від різних параметрів навчання може тривати години. На моїй локальній машині знадобилося близько 15 години, але перші позитивні результати можна було отримувати і за після 7 годин, тобто 4000 ітерацій.

Розроблена модель може ідентифікувати обличчя навіть на розмитому фоні. Було помічено, що стосовно людини, яка стоїть попереду, модель не настільки впевнена (38 % у чіткій області) у порівнянні з прогнозом для людини відразу за нею (100 % у розмитій області). Це може бути пов'язане з якістю

навчального набору даних, таким чином модель певною мірою схильна до впливу від якості зображення.

3.2.2. Навчання нейронної мережі

Для навчання нейронної мережі використовувалось бібліотека *Keras*. *Keras* – це *API* високого рівня, для створення глибокого машинного навчання написані на *Python* і здатні працювати з *TensorFlow*, *CNTK* та *Theano*. Це *API* було розроблено з можливим на швидке дії, можливість перейти від ідеї до результату з можливою затримкою. *Keras* використовують, якщо потрібна бібліотека для глибокого навчання, яка покращує легко і швидко взяти та прототипувати (завдяки зручності, модульності та розширюваності), можливе використання як згорткові мережі, так і повторюваних мережі, а також комбінації обох, безпечно працює на звичайному та графічному процесорі.

Нейронна мережа має 4 внутрішніх повнозв’язних шари по 256 нейронів у кожному, а вихідний шар має 32 нейрони.

Є можливість використати алгоритмічне розширення навчальних даних для покращення результатів, Простий спосіб збільшення навчальних даних – це можливе покращення кожного навчального зображення на один піксель вниз або наліво. Можемо зробити це, запустивши програму *expand_mnist.py* із підказки оболонки

$ *python expand_mnist.py*

Запуск цієї програми отримує доступ до 50 000 *MNIST* навчальних зображень, і готує розширений навчальний набір з 250 000 навчальних зображень. Ці дані потім можемо використовувати як навчальні зображення для мережі, але практика показала, о було достатньо і обраних файлів. У початкових експериментах було зменшено кількість епох тренувань – це мало сенс, оскільки тренуємо з ними в рази більше даних. На даний момент було використано 4100 зображень з відкритих джерел.

Всі упередження та ваги в об'єктах мережі можливе лише випадковим чином, з застосуванням функцію *Numpy np.random.randn* для генерації імовірнисних розподілів із середнім значенням 0 і таким відхиленням. Ця

випадкова ініціалізація дає алгоритму стохастичного градієнтного спуску місце, з якого слід почати.

Зміщення та ваги можуть бути представлені як списки матриць *Numpy*. Так, наприклад, *net.weights* [10] – матриця Нумпі, що містить ваги, що поєднують і третій та інші слої нейронів. Це така матриця w_{jk} – вага для такого між *k*-им нейроном у наступному шарі і *j*-им нейроном у третьому шарі.

Дані навчання – це лише кортежи (*x*, *y*), що можуть вхідні дані для навчання та відповідні такі результати. Змінні *epochs* і *mini_batch_size* – кількість епох, для яких потрібно навчатися, і розмір міні-партій, які слід лише мати під час вибірки. *eta* – швидкість тренування (*η*). Якщо наведено опціональний аргумент *test_data*, програма буде вирішувати мережу після кожної епохи навчання та роз'ясняти частковий прогрес.

Код працює наступним чином. Таким чиом він починається з імовірнісного перемішування навчальних даних, а потім розподіл їх на партії за певного розміру. Це простий спосіб можливої вибірки з навчальних даних. Потім для таких і інших *mini_batch* застосовуємо один лише градієнтний алгоритм. Це робиться з можливістю коду *self.update_mini_batch* (*mini_batch, eta*), який оновлює ваги та можливості мережі відповідно до такої ітерації градієнтного алгоритму, використовуючи лише такі дані в *mini_batch*. Ось код для методу *update_mini_batch*:

Це активує алгоритм зворотного розповсюдження, що є швидким способом вирахування градієнта функції витрат. Отже *update_mini_batch* працює просто, обчислюючи ці градієнти для кожного навчального прикладу в *mini_batch*, а потім належним чином оновлюючи *self.weights* і *self.biases* (рис. 3.7).

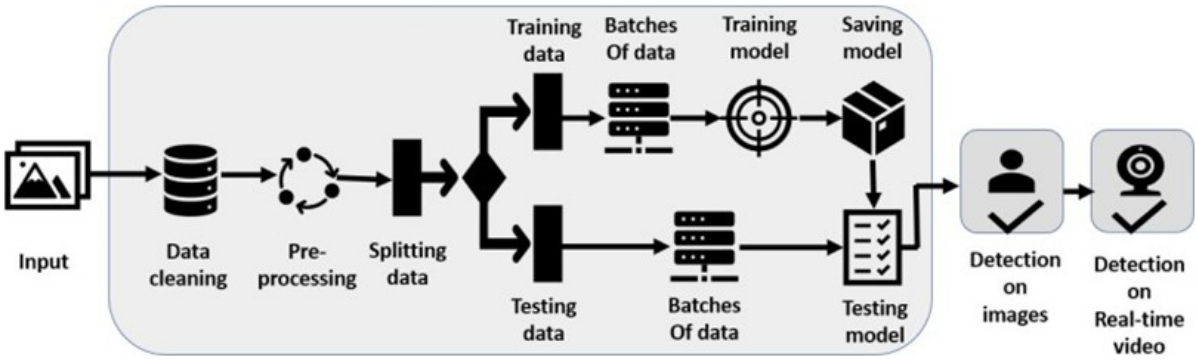


Рис. 3.7. Схема роботи моделі

3.2.3. Класифікація зображень за допомогою *MobileNetV2*

MobileNetV2 – це глибинна нейронна мережа, яка була розгорнута для вирішення проблеми класифікації. Попередньо навчені ваги *ImageNet* були завантажені з *TensorFlow* (рис. 3.8).

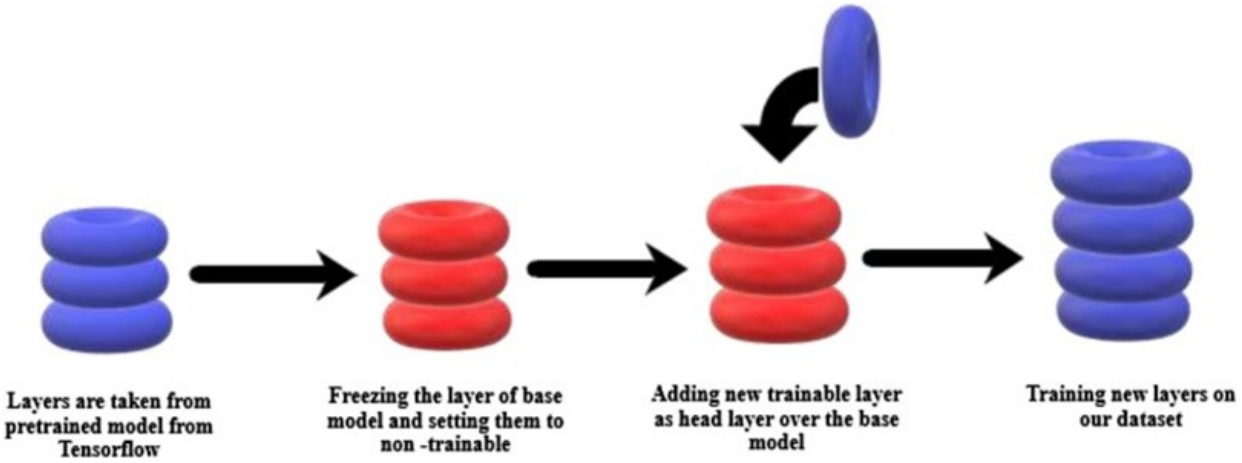


Рис. 3.8. Конвеєр використання попередньо підготовленої моделі.

Потім базові шари заморожуються, щоб уникнути погіршення вже вивчених функцій. Потім додаються нові шари, які можна навчати, і ці шари навчаються на зібраному наборі даних, щоб він міг визначити ознаки для класифікації обличчя в масці від обличчя без маски. Потім модель доопрацьовується, а потім ваги зберігаються. Використання попередньо навчених моделей допомагає уникнути непотрібних обчислювальних витрат і допомагає скористатися перевагами вже зміщених ваг без втрати вже вивчених функцій.

Архітектура *MobileNetV2* представлена на рис. 3.9.

Згортковий шар є фундаментальним блоком згорткової нейронної мережі. Термін згортка означає математичну комбінацію двох функцій для отримання третьої функції. Він працює на механізмі розсувного вікна, що допомагає витягувати особливості із зображення. Це допомагає генерувати карти об’єктів. Згортка двох функціональних матриць, одна з яких є матрицею вхідного зображення *A*, а інша є згортковим ядром *B*, дає нам вихід *C* як:

$$C(T)=(A*B)(x)=\int A(T)\times B(T-x)\,dT \qquad (3.1)$$

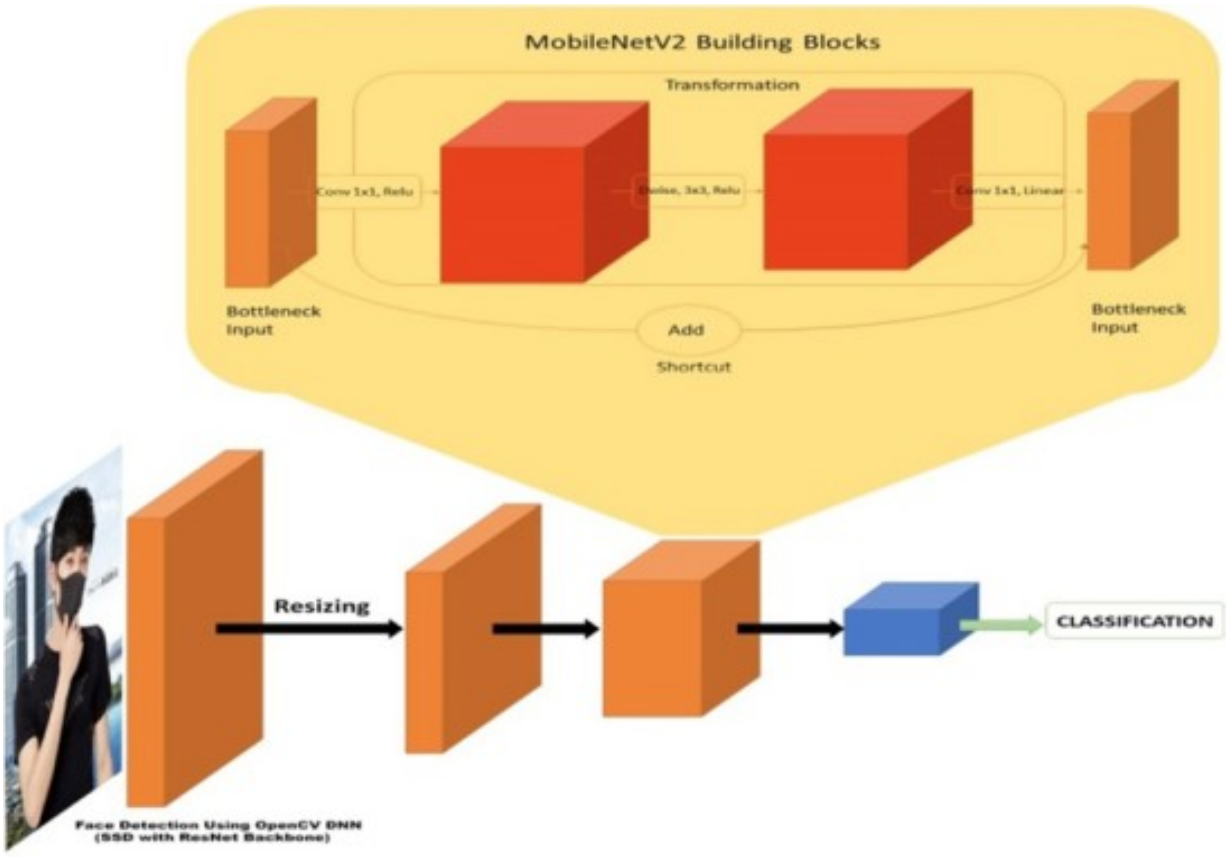


Рис. 3.9. Архітектура *MobileNetV*2

Застосування операцій об’єднання може прискорити обчислення, дозволяючи зменшити розмір вхідної матриці без втрати багатьох функцій. Можуть застосовуватися різні види операцій об’єднання:

– максимальний пул: бере максимальне значення, наявне у вибраному регіоні, де зараз знаходиться ядро, як значення для вихідного значення матриці для цієї комірки;

– середній пул: він приймає середнє значення всіх значень, які в даний момент знаходяться в регіоні, де знаходиться ядро, і приймає це значення як вихідне значення для значення матриці цієї комірки. Рис. 3.10 показує операцію середнього об’єднання.

Випадаючий шар допомагає зменшити переобладнання, яке може виникнути під час навчання, відкидаючи випадкові упереджені нейрони з моделі. Ці нейрони можуть бути частиною прихованих шарів, а також видимих шарів. Імовірність випадання нейрона можна змінити, змінивши коефіцієнт випадання.

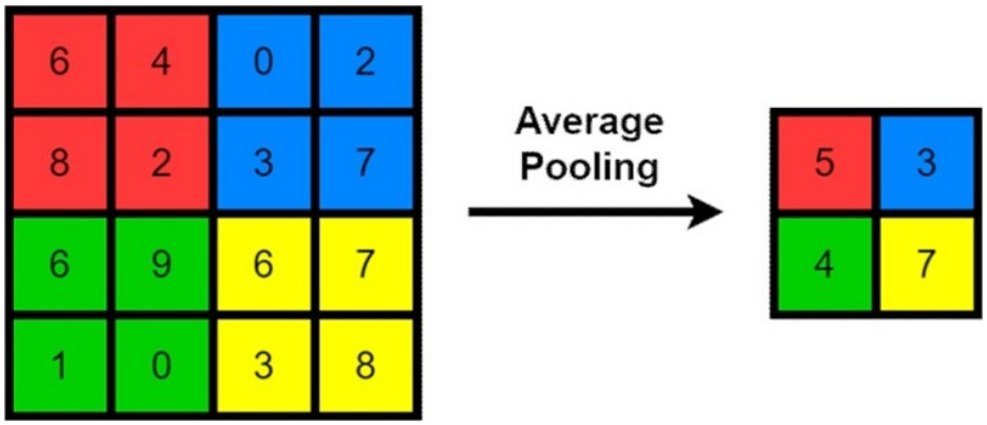


Рис. 3.10. Операція об’єднання середнього значення

Нелінійний шар, зазвичай, розташовується за згортковими шарами. Найчастіше використовувані нелінійні функції включають різні види випрямленої лінійної одиниці (*ReLU*) [9], *Leaky ReLU* [11], *Noisy ReLU* [13], *Exponential ReLU* [14] тощо, сигмовидна функція, а також функції *tanh*. Нижче показано різні види нелінійних функцій та їх рівняння.

$$Sigmoid: \sigma(x) = 11+e^{-x} \quad (3.2)$$

$$Leaky\ Relu: f(x)=max(0.1x, x) \quad (3.3)$$

$$Tanh: f(x)=tanh(x) \quad (3.4)$$

$$Maxout: f(x)=max(wT1x+ b1 , wT2x+b2) \quad (3.5)$$

$$Relu: f(x)=max(0,x) \quad (3.6)$$

$$ELU: f(x)= \{x\ x\geq 0\ \alpha(ex-1)\ x<0\ ) \quad (3.7)$$

Повністю підключені шари додаються в модель і мають повні зв’язки з шарами активації. Ці шари допомагають класифікувати дані зображення в багатокласовій або бінарній класифікації. *SoftMax* є прикладом функцій активації, які використовуються на цих рівнях, і він надає результат прогнозованих вихідних класів з точки зору ймовірності.

Оскільки численні множення матриці не можуть бути зменшені до однієї числової операції, нелінійні функції активації, такі як *ReLU*6, застосовуються в нейронних мережах, тому кілька розбіжностей можна легко видалити. Завдяки цьому можна побудувати багатошарову нейронну мережу. Оскільки функція активації *ReLU* відмовляється від значень, менших за 0. Розміри мережі

збільшуються за рахунок збільшення кількості каналів для боротьби з втратою інформації.

Для перевернутого залишкового блоку шари блоків стискаються, і виконується протилежна процедура, як зроблено вище. Це робиться в точці, де пов'язані з'єднання пропуску; це може вплинути на роботу мережі. Щоб впоратися з цим, було введено концепцію лінійного вузького місця, в якому перед додаванням блоку до початкової активації останній згорток блоку, що залишився, отримує лінійний вихід.

3.2.4. Алгоритми повного конвеєру

Запропонована методологія може бути пояснена за допомогою двох алгоритмів, як показано нижче. Спочатку зображення були попередньо оброблені та навчені для всього набору даних. Потім модель, навчена в першій частині, була використана для визначення маски для обличчя з відповідною точністю. У алгоритмі на рис. 3.11 зображення разом із значеннями пікселів були взяті як вхідні дані, змінено їх розмір та нормалізовано.

Потім до зображень було застосовано техніку збільшення даних, щоб отримати більшу точність. Потім дані були поділені на навчальні та тестові пакети, і до них застосували модель *MobilenetV*2. Для компіляції всієї моделі використовувався оптимізатор Адама.

В алгоритмі на рис. 3.12 модель, навчена в попередній частині, була потім розгорнута як для класифікації на статичних зображеннях, так і на веб-камері реального часу.

Використання зображень з спеціальними маскувальними елементами (рис. 3.13) погіршувала якість визначення медичних масок на обличчі в цілому, тому біло вирішено відкотити навчальну модель о попереднього стану.

Припущенням погіршення результатів є внесення багатоваріантності при ідентифікації обличь та їх контурів на зображеннях з маскуючими елементами.

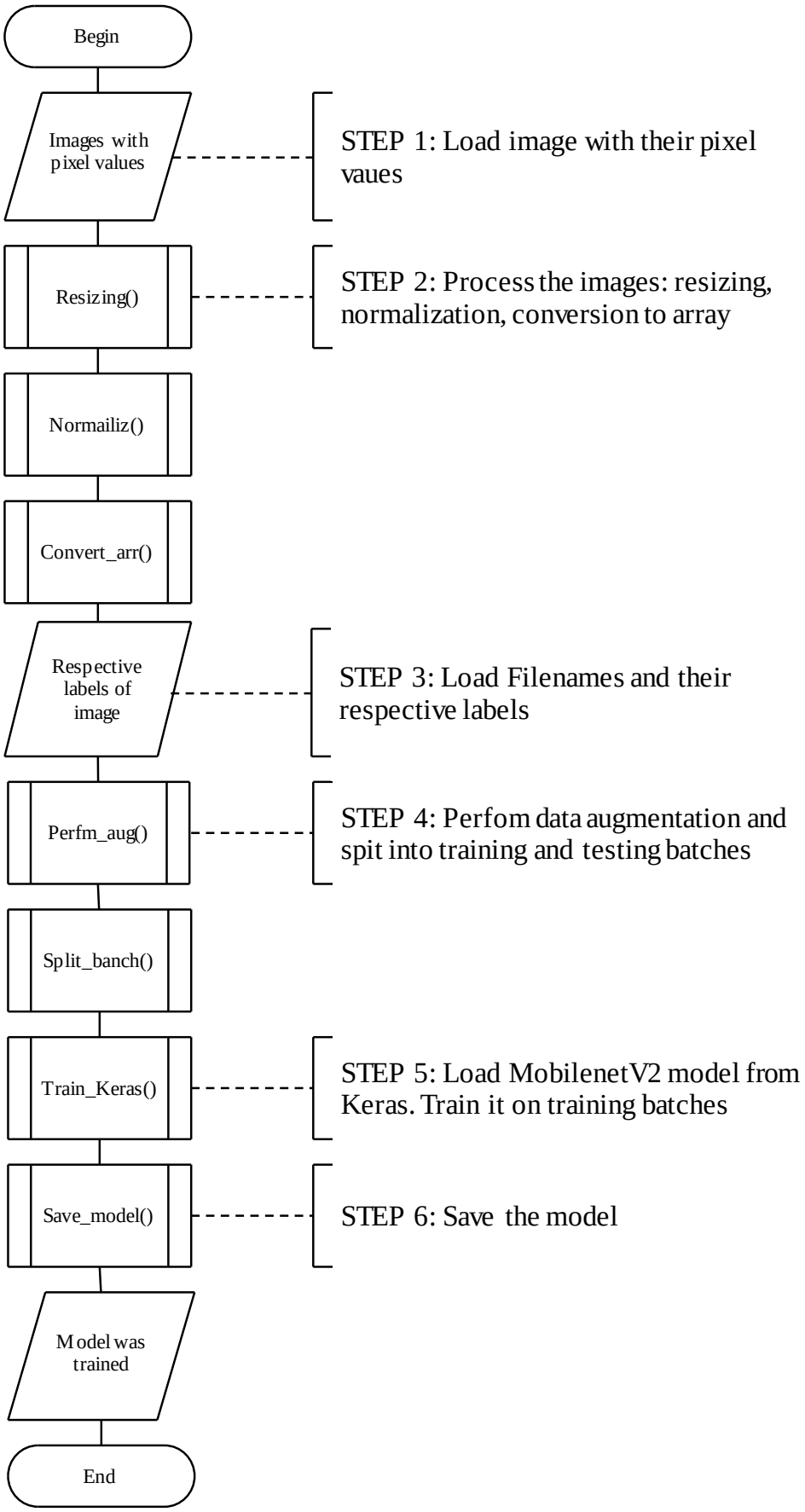


Рис. 3.11. Схема алгоритму попередньої обробки і тренування моделі

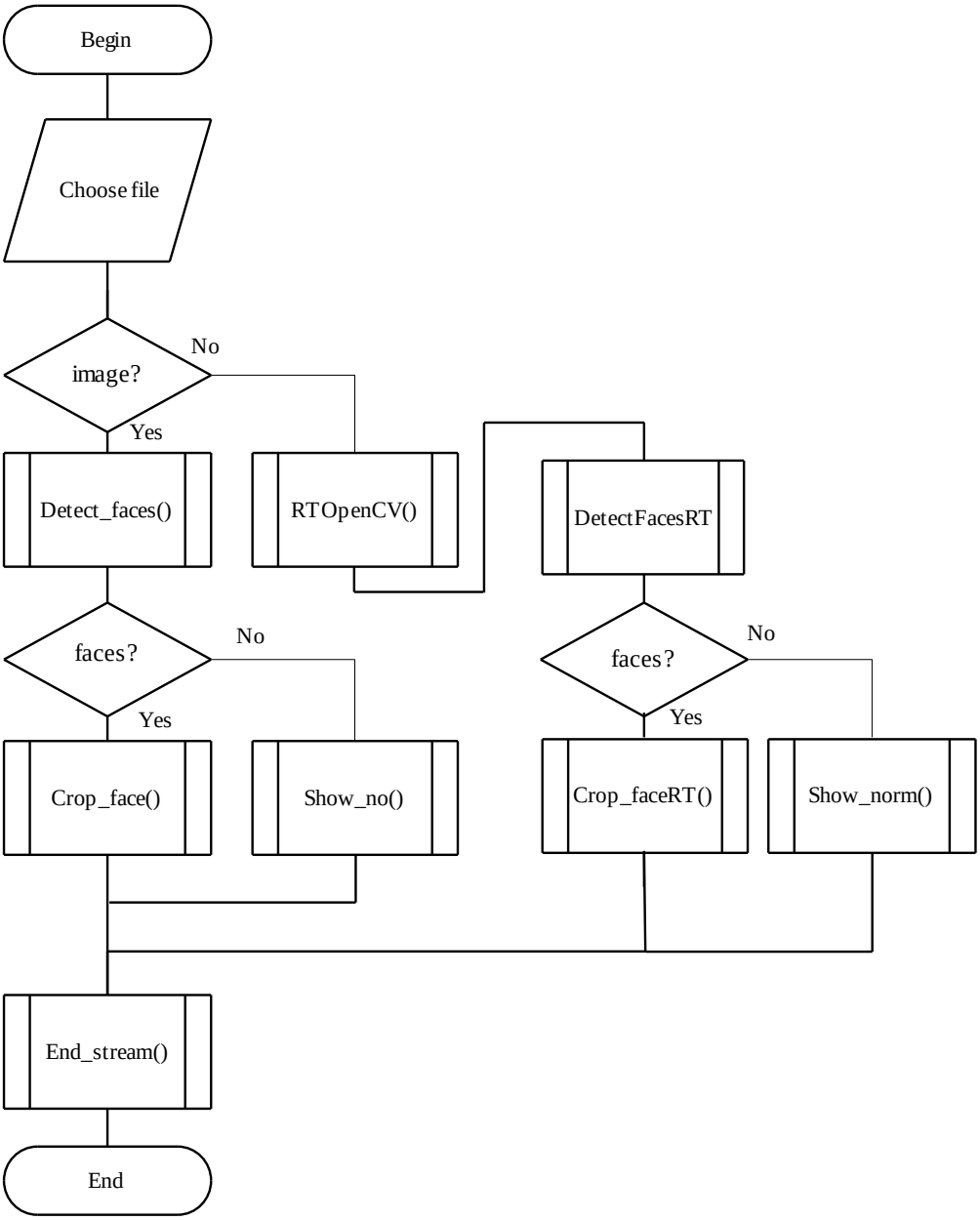


Рис. 3.12. Схема алгоритму визначення медичних масок на обличчях у зображеннях і у відеофайлах

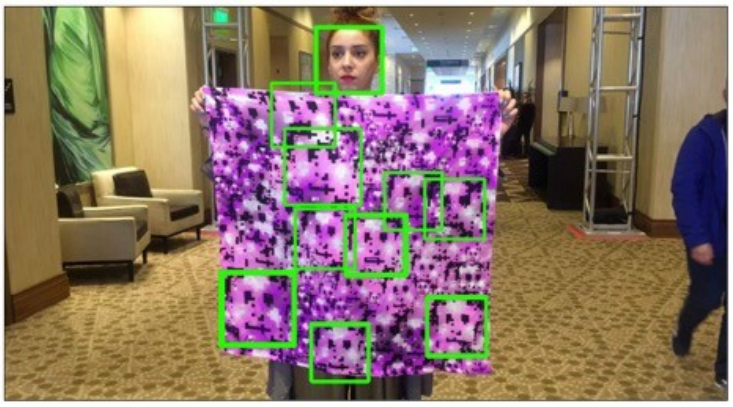


Рис. 3.13. Приклад зображення з маскуючими елементами

3.3. Реалізований віконний інтерфейс

На основі діаграма послідовності дій для запису даних в системі, яку було представлено в розділі 3.1 на рис. 3.2 було створено схему алгоритму роботи основного вікна системи визначення у зображеннях медичних масок на обличчях (рис. 3.14).

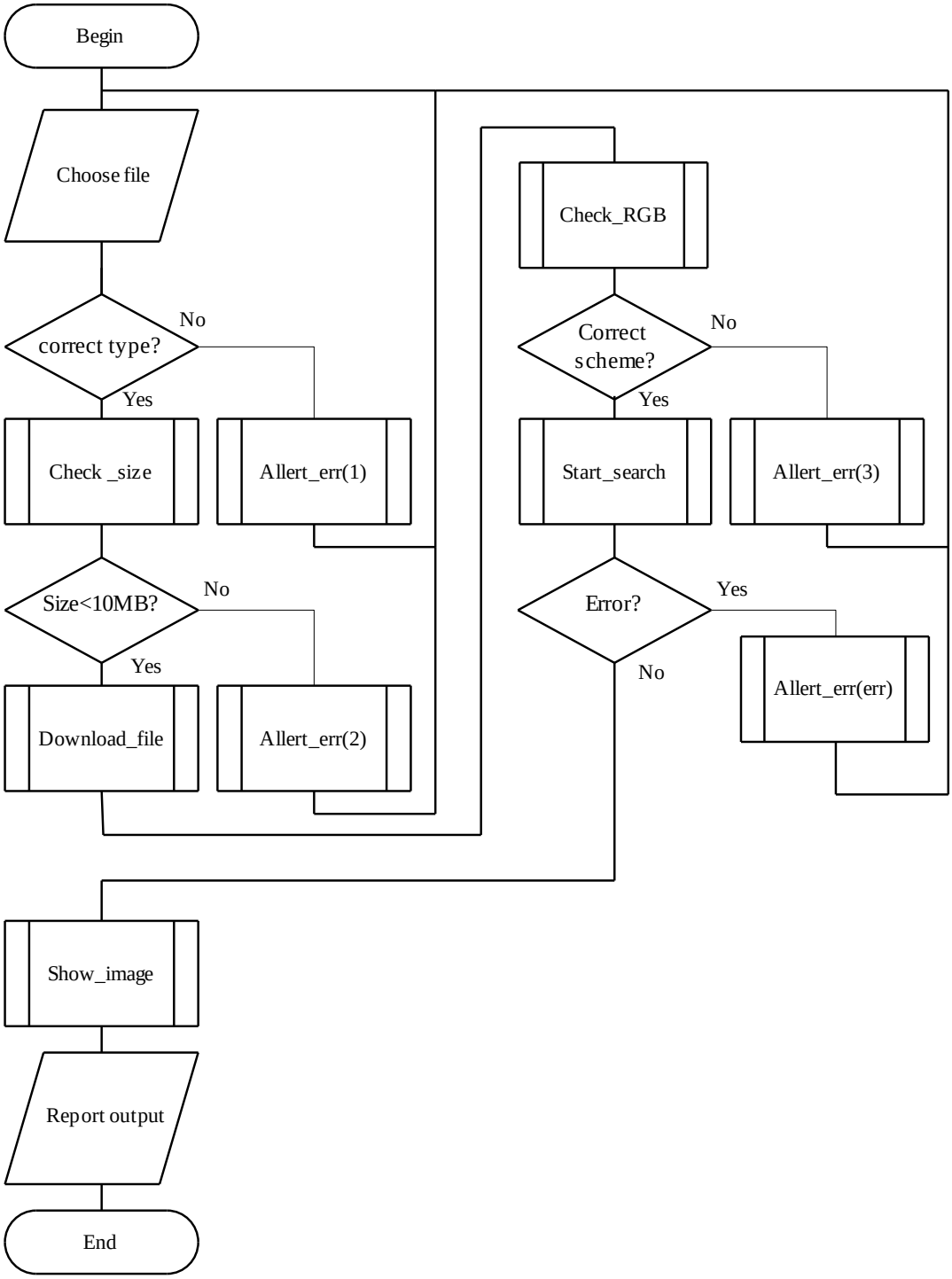


Рис. 3.14. Схема алгоритму роботи основного вікна системи визначення медичних масок на обличчях у зображеннях

У відповіності до цього алгоритму було реалізовано всі етапи обробки зображень і нанесення на них позначок про наявність і відсутність медичних масок. Даний модуль написано також мовою *Python*, як і весь бекенд системи.

На рис. 3.15 представлено розроблений віконний інтерфейс з результатом завантаження зображення ля подальшого аналізу.

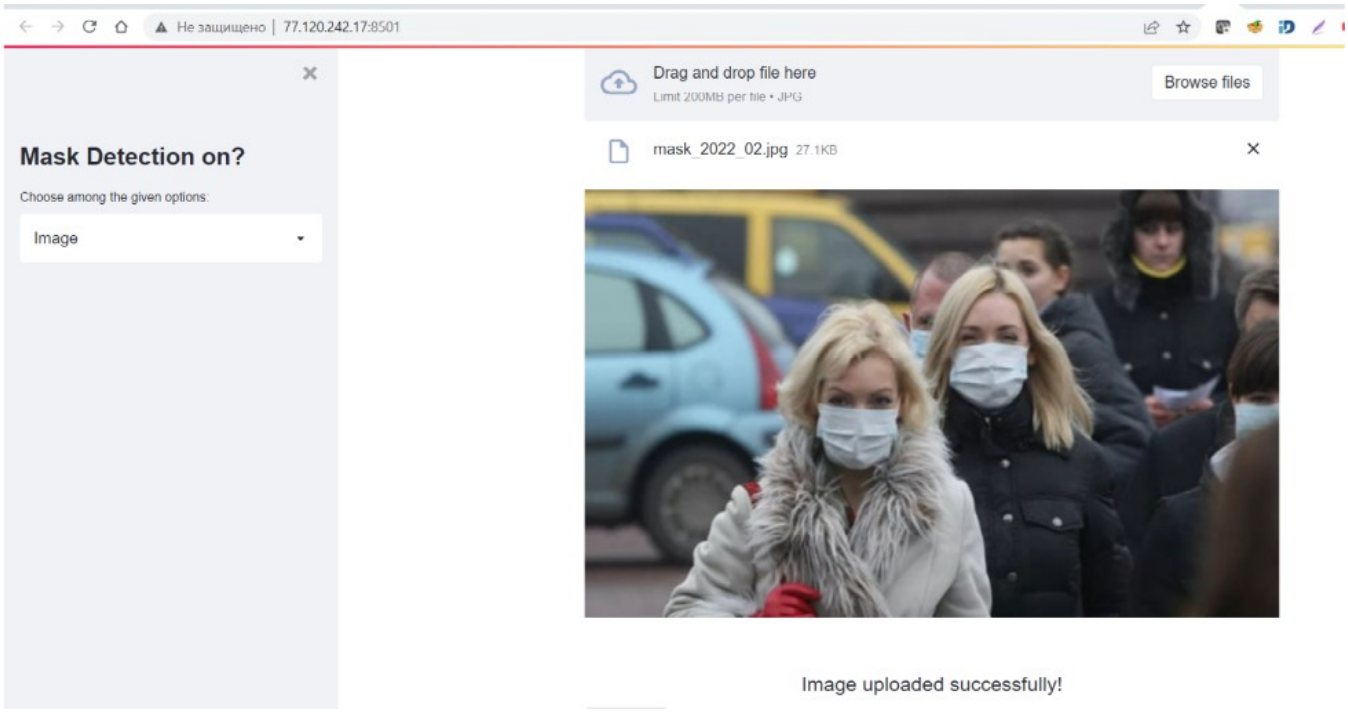


Рис. 3.15. Вікно системи з результатом завантаження зображення

На рис. 3.16 представлено нанесені позначки про наявність масок на обличчах і результати оцінки достовірності прийнятого рішення.

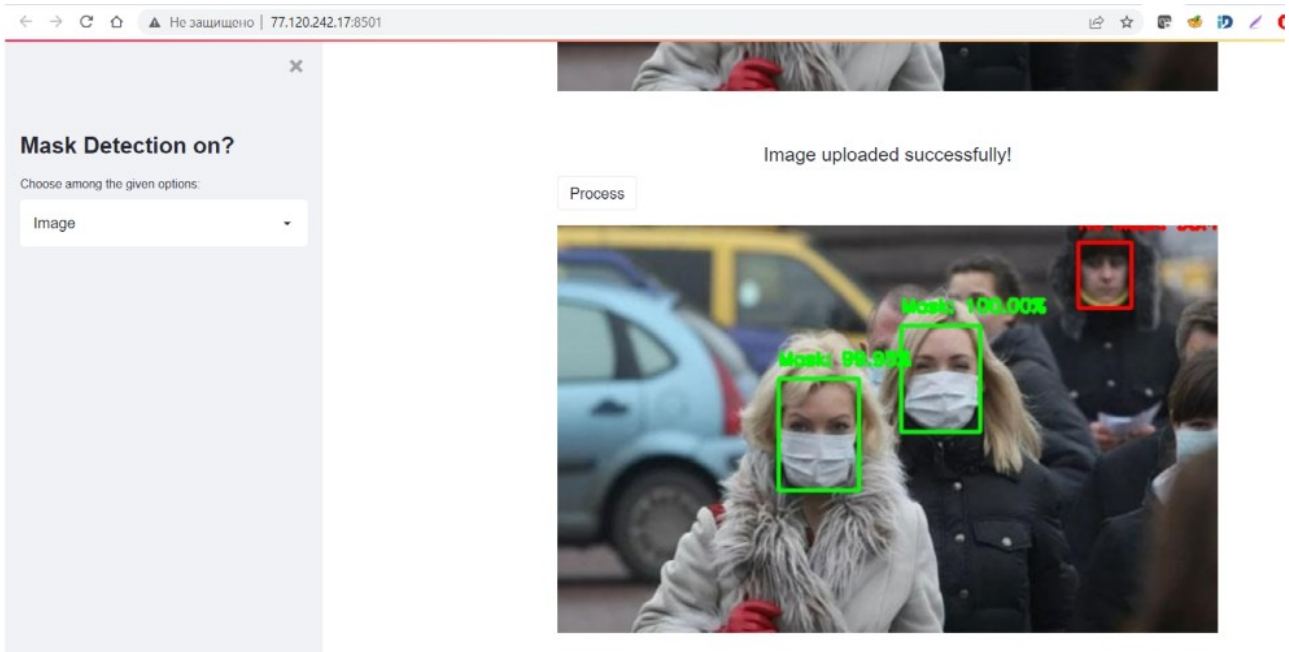


Рис. 3.16. Вікно системи з позначками про наявність масок на обличчах

В ході розробки програмного інтерфейсу було внесено зміни, які стосувались розміру і типу файлів. Це було обумовлено вкликим часом обробки зображень розміром більше 10 МБ.

Розроблений інтерфейс передбачає гнучкість і адаптивність, о озволяє празвати з зображеннями у різній орієнтції (рис. 3.17). На рис. 3.18 відзначено обличчя без маски.

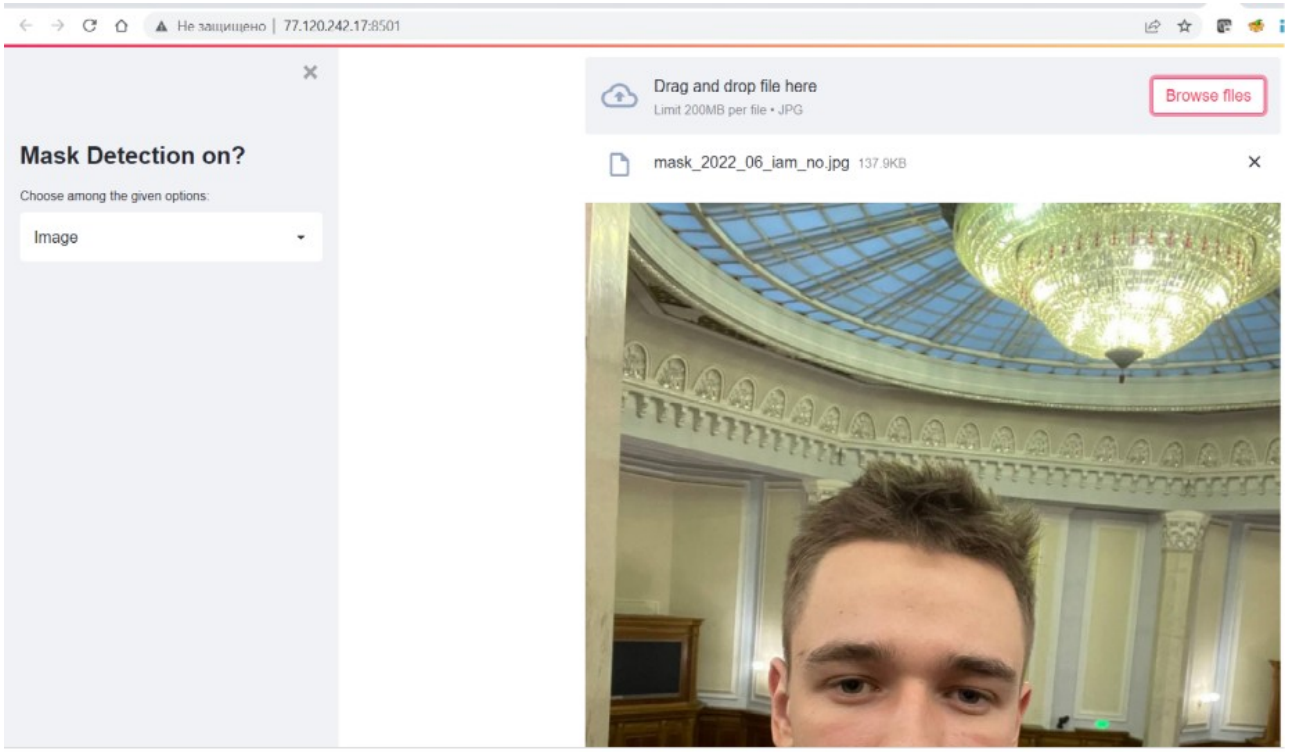


Рис. 3.17. Вікно системи з вертикально орієнтаціє зображення

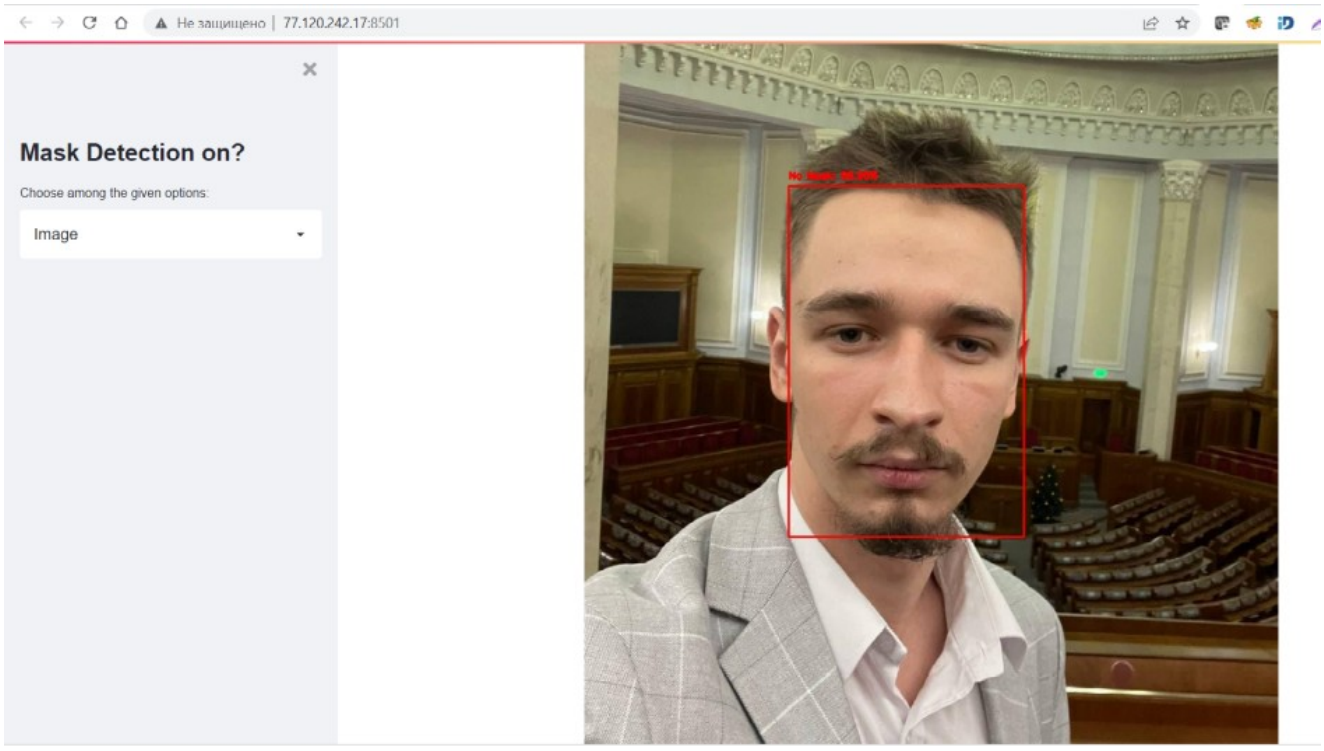


Рис. 3.18. Вікно системи з знайденним обличчям без маски

На рис. 3.19 відзначено обличчя в масці. Але, як бачимо за результатом, то інші обличчя взагалі не було відокремлено. Це пов’язано з недостатнім набором обличь у профіль для навчання моделі.

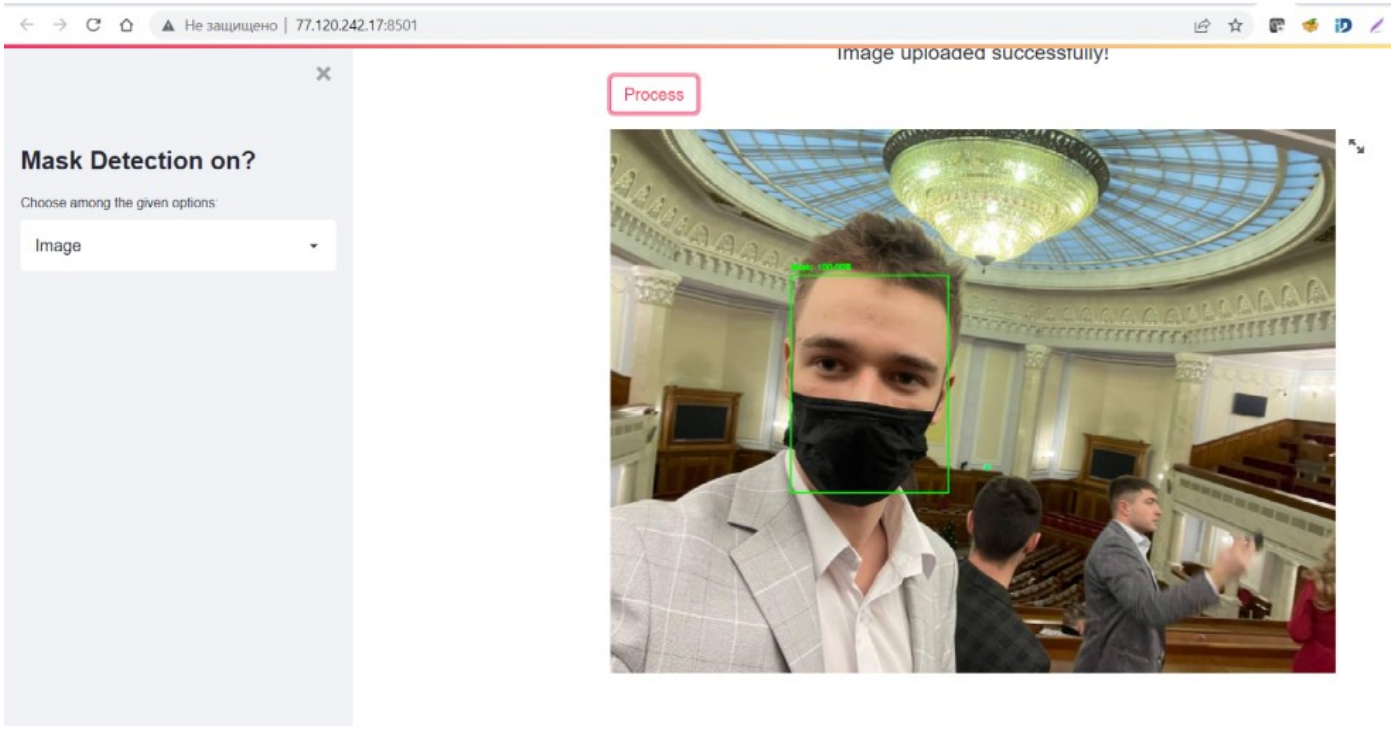


Рис. 3.19. Вікно системи з помилкою у відокремленні обличь

На рис. 3.20 наведено приклад визначення обличчя в масці на зображенні з засвіченим фоном.

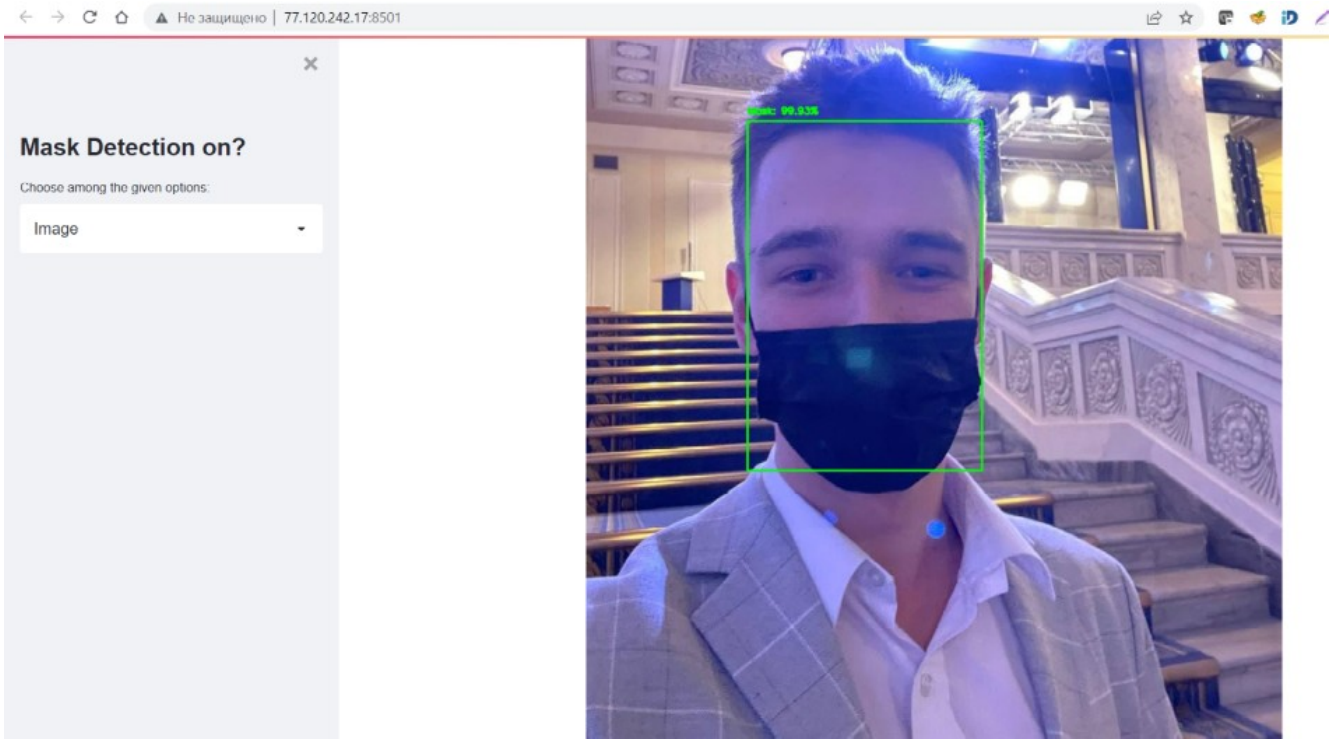


Рис. 3.20. Приклад роботи системи з нечіткими зображеннями

3.4. Експериментальні дослідження розробленої системи

Усі експериментальні випробування проводилися на ноутбуці з процесором *Intel i*7-8750*H* (4,1 ГГц), 16 ГБ оперативної пам’яті та 1050*ti max-Q* з 4 ГБ відеопам’яті. Програмне забезпечення *Jupyter Notebook*, оснащене ядром *Python* 3.8, було обрано в цьому дослідженні для розробки та впровадження різних експериментальних шляхів.

Показники, вибрані для оцінки моделі, пояснюються нижче.

$$Accuracy=Tp+Tn(Tp+Fp+Fn+Tn) \quad (3.8)$$

$$Precision=Tp(Tp+Fn) \quad (3.9)$$

$$Recall=Tp(Tp+Fn) \quad (3.10)$$

$$f1\ score=2*Recall*Precision(Recall+Precision) \quad (3.11)$$

де *Tp – True positive,*

Tn – True negotive,

Fp – False positive,

Fn – False negotive.

У вищенаведених формулах *True positive* значення відносяться до зображень, які були позначені як істинні і після передбачення моделлю дали істинний результат. Аналогічно, для *True negative* значення відноситься до зображень, які були позначені як істинні, але після передбачення результату був хибний. *False positive* відноситься до зображень, які були позначені як хибні і після передбачення призвели до помилкових, отже, хибних позитивних результатів. *False negative* відноситься до зображень, які були позначені як хибні і після передбачення привели до істинних, отже, помилкових негативів. Точність дала міру позитивних прогнозованих значень. Відкликання дало можливість класифікатору знайти всі позитивні зразки, а оцінка *f*1 дала міру точності тесту. Ці показники оцінки були обрані через їхню здатність давати найкращі результати у збалансованому наборі даних.

По-перше, для сортування списку в лексикографічному порядку використовується функція з назвою *sorted_* алфавітно-цифровим. Визначено функцію попередньої обробки, яка переносить папку в набір даних як вхідні дані,

потім завантажує всі файли з папки та змінює розміри зображень відповідно до моделі. Потім після сортування списку зображення перетворюються в тензори. Потім список перетворюється на масив *NumPy* для швидшого обчислення. Після застосування попередньої обробки точність нашої моделі значно зросла. Після цього застосовується процес збільшення даних для підвищення точності після навчання моделі.

Наведені результати моделі до розширення даних (рис. 3.21 та 3.22). Модель без аугментації даних показала значне зниження її зростання, як і при навчанні через 100 епох після збільшення даних. Відображаючи точність навчання без збільшення даних, модель намагалася вивчати особливості до 60 епох у точності перевірки, а потім стала стабільною.

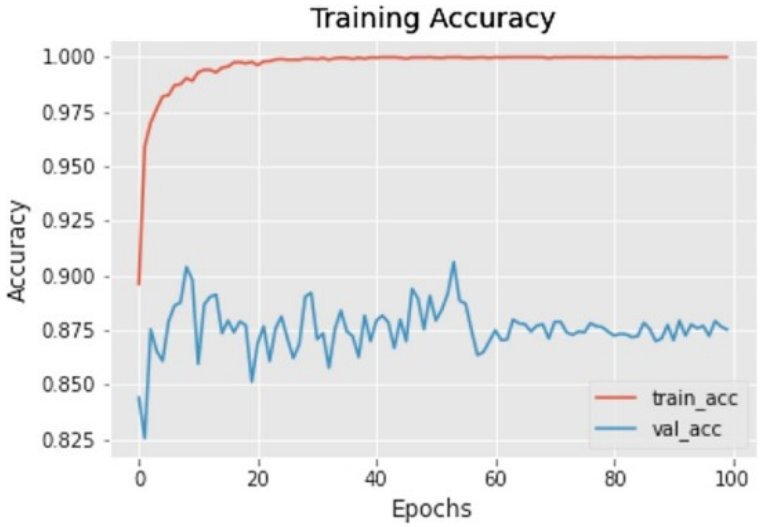


Рис. 3.21. Крива точності навчання (без збільшення даних)

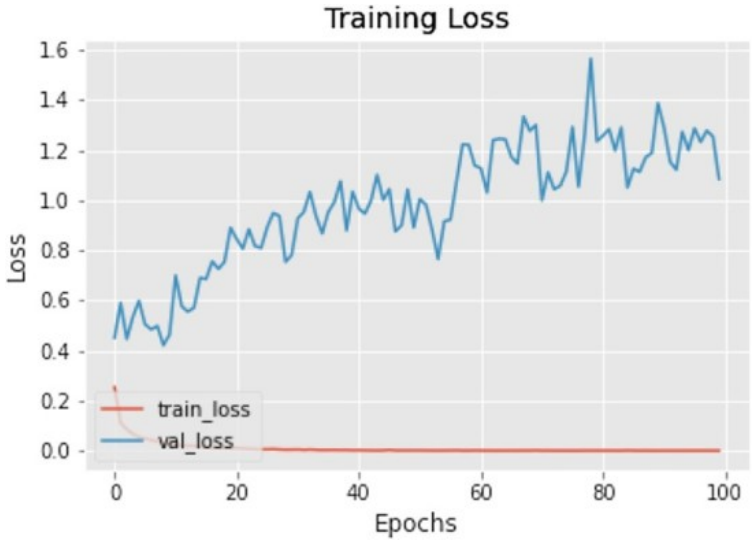


Рис. 3.22. Крива втрат на тренуванні (без збільшення даних)

Після 100 епох точність навчання виявилася 87,51 % у порівнянні з 92,64 % точністю навчання при застосуванні додавання даних. На рис. 3.21, що зображує втрату спроможності до навчання на наборі без збільшення даних, показано втрату навчання моделі до 10-ї епохи і починалася з приблизно 0,5 втрати при втраті перевірки, максимальна втрата підтвердження на піку 1,6 за 80 епох.

Рис. 3.23 та 3.24 показують покращення характеристик після застосування розширення даних, яке неможливо було досягти до збільшення даних.

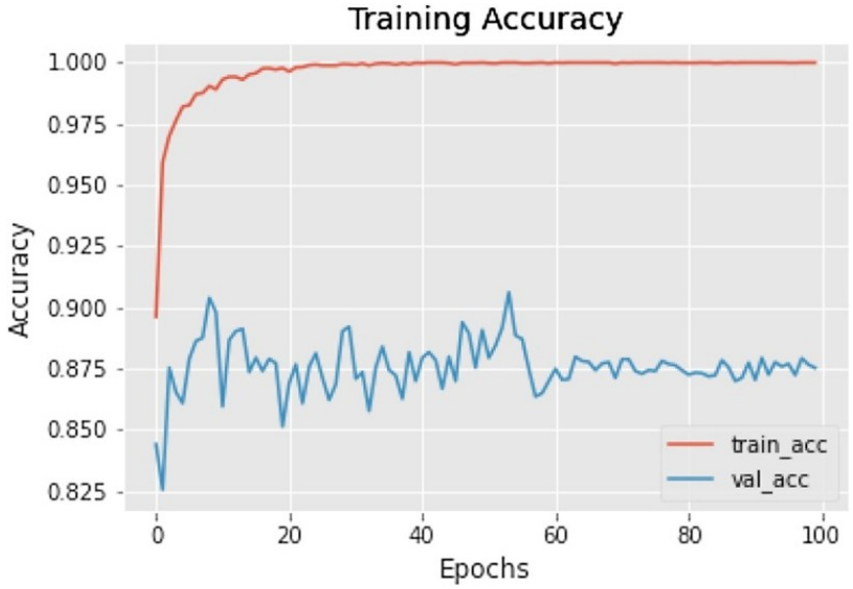


Рис. 3.23. Крива точності навчання

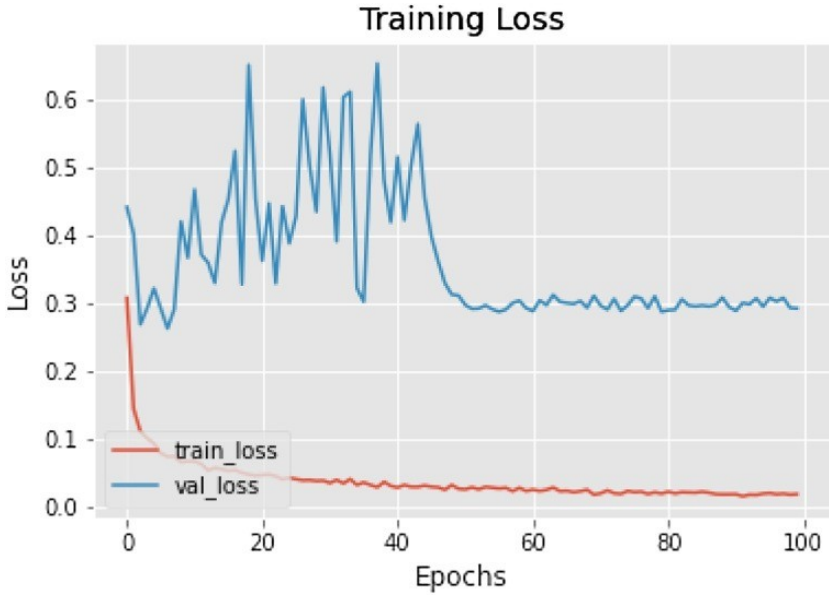


Рис. 3.24. Крива втрат на тренуванні

Для розв’язання бінарної класифікації в роботі використовується проблема моделі глибокого навчання. *Keras* використовується для створення моделі класифікації, яка є *API* штучних нейронних мереж розширеного рівня. Точність дає нам рівень правильного прогнозування людини в масці, ідентифікованого машиною за запропонованою моделлю. В рис. 3.23 зображуючи точність навчання з розширенням даних, модель намагалася вивчати особливості до 42 епох у точності перевірки, а потім стала стабільною. Через 100 епох точність навчання склала 92,64%. Середня точність моделі становить «93%» для передбачення того, чи носить людина маску чи ні в наборі даних перевірки. Червона крива показує точність навчання, яка майже дорівнює 99 %, тоді як синя лінія представляє точність набору даних перевірки. Крива втрат при навчанні, що відповідає навчанню та підтвердженню, показана на рис. 3.24. Тут червона лінія показує втрату в наборі даних для навчання менше 0,1, тоді як синя лінія зображує втрату навчання в наборі даних перевірки.

3.5. Висновки до розділу 3

Проведено проектування ПЗ та реалізовано основні класи та методи для визначення наявності медичної маски на обличчі.

Було розроблено та програмно реалізовано алгоритми повного конвеєру розпізнавання образів.

Розроблена модель може ідентифікувати обличчя навіть на розмитому фоні, і це викликає захоплення. Було помічено, що стосовно людини, яка стоїть попереду, модель не настільки впевнена (38 % у чіткій області) у порівнянні з прогнозом для людини відразу за нею (100 % у розмитій області). Це може бути пов'язане з якістю навчального набору даних, таким чином модель певною мірою схильна до впливу від якості зображення.

Середня точність моделі становить «93%» для передбачення того, чи носить людина маску чи ні в наборі даних перевірки.

РОЗДІЛ 4
SOFTWARE ARCHITECTURE DOCUMENT

Taras Shevchenko National University of Kyiv

A MASKED-FACE RECOGNITION SOFTWARE

Software Architecture Document (SAD)

Content owner: Semyon YAKIMOVICH

REVISION HISTORY

DOCUMENT NUMBER:	RELEASE/REVISION:	RELEASE/REVISION
1	V1.1	DATE: 2022/04/04

1. Documentation Roadmap

1.1. Document Management and Configuration Control Information

1.2. Purpose and Scope of the SAD

1.3. Viewpoint Definitions

1.3.1. Viewpoint Definition

1.3.1.1. Abstract

1.3.1.2. Stakeholders and Their Concerns Addressed

1.3.1.3. Elements, Relations, Properties, and Constraints

1.3.1.4. Language(s) to Model/Represent Conforming Views

2. Architecture Background

2.1. Problem Background

2.1.1. System Overview

2.1.2. Goals and Context

2.1.3. Significant Driving Requirements

2.2. Solution Background

2.2.1. Architectural Approaches

2.2.2. Analysis Results

2.2.3. Requirements Coverage

3. Referenced Materials

4 Directory

4.1. Index

4.2. Glossary

4.3. Acronym List

5. Sample Figures & Tables

1. Documentation Roadmap

1.1. Document Management and Configuration Control Information

- Revision Number: v1.1
- Revision Release Date: 2022/04/04
- Purpose of Revision: -
- Scope of Revision: -

1.2. Purpose and Scope of the SAD

Pattern recognition is one of the important problems in the theory of intelligent systems. On the other hand, the task of image recognition is of great advanced practical usage.

Face recognition tasks have been solved for over 40 years. These include:

- search and recognition of several people in the stream;
- resistance to changes in the face, hair, beard, glasses, age and face rotation;
- scalability of data for face identification;
- real time processing .

The task of recognizing faces and objects on the face (for example, medical masks) is part of the practical application of the theory of pattern recognition. It consists of two subtasks: identification and classification. Identification is actively used in modern services such as iPhoto. Face recognition is used everywhere, from FaceID to iPhone X, to use in targeting military equipment.

To implement object localization by software, it is necessary to develop a set of interconnected algorithms and modules that can retrieve images from an external device or file, pre-process it, localize all possible or conditional rules of the object group system.

The software is written in Python using open source technology solutions and

frameworks. The approach proposed in the thesis uses deep learning, TensorFlow, Keras and OpenCV. The developed model can be used for security and information purposes. The developed approach involves selecting individual images from streaming video or obtaining individual images to identify the face using the MobilenetV2 architecture classifier.

The developed software does not use any modified data sets with masked images, which was due to the deterioration of the model when introducing additional training sets with masked images.

The basic data set consists of 4100 images, which can be divided into two classes: 2165 images with a mask and 1935 images without a mask. The images used were obtained from the following open sources: Bing Search API, Kaggle, RMFD.

Achieved accuracy of face detection with mask for images near of the 93%. The great influence of image quality on the definition of masked faces was determined.

1.3. Viewpoint Definitions

The software is written in Python using open source technology solutions and frameworks. The approach proposed in the thesis uses deep learning, TensorFlow, Keras and OpenCV. The developed model can be used for security and information purposes. The developed approach involves selecting individual images from streaming video or obtaining individual images to identify the face using the MobilenetV2 architecture classifier.

The developed software does not use any modified data sets with masked images, which was due to the deterioration of the model when introducing additional training sets with masked images.

2. Architecture Background

2.1. Problem Background

Detecting a face mask has proved to be an amazing problem in the field of image processing and computer vision. Face recognition has a variety of uses, ranging from face recognition to fixing facial movements, where the latter requires the disclosure of the face with very high accuracy. Due to the rapid progress in the field of machine learning algorithms, the problems associated with the technology of detecting face masks are quite well solved. This technology is more relevant today because it is used to detect faces not only in still images and videos, but also in real time. With the achievement of convolutional neural networks [1] and deep learning [2], very high accuracy in image classification and object detection can be achieved.

Due to the sudden onset of the COVID-19 pandemic, various face recognition technologies are now available to people in masks. HanvonTechnologyWang [4] reported that the accuracy of face recognition in the mask is about 85%. Accuracy over 90% was obtained from Minivision Technology [5]. The multifaceted model based on the face and eyes [7] achieves 95% recognition accuracy. Wang, Lee, and Fei (2020) used the YOLOv3 algorithm to detect a face mask [8]. This method reached 93.9% accuracy.

2.1.1. System Overview

The main essences of the system are:

– AppUser – the basic essence of the user;

– Administrator – extension of the user entity to define administrator rights;

– Image – the basic essence of the image;

– Analyzer – image analysis module;

– SearchInfo – search module for relevant classes in the database;

– Training – model training module;

– OpenCV, ResNet50_v2 – external libraries.

2.1.2. Goals and Context

The purpose of the study is to develop software to ensure the recognition of a person in a mask.

The object of research - recognition of objects on static images.

The subject of research - Software for recognizing a person in a mask.

2.1.3. Significant Driving Requirements

Face mask medical detection software implements face recognition on static images and allows you to identify them by the presence of a medical mask.

From this point of view, the system can be represented by the following diagram of precedents (Fig. 3.1). This diagram depicts the relationship between actors and precedents in the system.

The system provides 2 groups of actors, each of which is provided with different levels of access:

1) administrator - a role that allows not only to fully use the system, but also can start the learning process;

2) user - a role that is provided to all users of the system without registration. This role has access to upload and analyze images.

2.2. Solution Background

Pattern processing is one of the most fundamental problems in the theory of intelligent systems. On the other hand, the task of pattern recognition is of great practical usage.

Face recognition tasks have been solved for over 40 years. These include:

- search and recognition of several people in the stream;
- resistance to changes in the face, hair, beard, age and face rotation;
- scalability of data for face identification;
- real time processing .

The task of recognizing faces and objects on the face (for example, medical masks) is part of the practical application of the theory of pattern recognition. It

consists of two subtasks: identification and classification. Identification is actively used in variety of services such as Facebook, iPhoto. Face recognition is used everywhere, from FaceID to iPhone X, to use in targeting military equipment.

To implement the localization of objects by software it is important to develop a set of interconnected algorithms and modules that can retrieve images from an external device or file, pre-process it, localize all possible or due to system rules object group.

2.2.1. Architectural Approaches

To begin with, to make a facemask detector, you need the appropriate data. In addition, due to the nature of the external library, annotated data with limiting rectangles is required. One option is to create very own dataset, either by collecting images from the Internet, or by taking photos of friends, acquaintances, and annotating photos manually using certain programs, such as*LabelImg*.

The basic data set consists of 4100 images, which can be divided into two classes: 2165 images with a mask and 1935 images without a mask (Fig. 3.4). The images used were obtained from the following open sources: Bing Search API, Kaggle, RMFD.

The Keras library was used to train the neural network. Keras is a high-level API for creating deep machine learning written in Python and capable of working with TensorFlow, CNTK and Theano.

MobileNetV2 is a deep neural network that has been deployed to solve the classification problem. Previously trained ImageNet scales were loaded from TensorFlow (Fig. 3.8).

The base layers are then frozen to avoid deterioration of the functions already studied. New, teachable layers are then added, and these layers are trained on the collected data set so that it can identify features for classifying a face in a mask from a face without a mask. Then the model is refined, and then the scales are saved. The use of pre-trained models helps to avoid unnecessary computational costs and helps to take advantage of already shifted weights without losing already learned functions.

The MobileNetV2 architecture is presented in Fig. 3.9.

2.2.2. Analysis Results

The basic data set consists of 4100 images, which can be divided into two classes: 2165 images with a mask and 1935 images without a mask. The images used were obtained from the following open sources: Bing Search API, Kaggle, RMFD.

Achieved accuracy of face detection with mask for images near of 93%. The great influence of image quality on the definition of masked faces was determined.

2.2.3. Requirements Coverage

All experimental tests were performed on a laptop with Intel i7-8750H (4.1 GHz), 16 GB of RAM and 1050ti max-Q with 4 GB of video memory.

The following components must be installed:TensorFlow, Keras, OpenCV.

3. Referenced Materials

1. Lowrence S., Giles C.L., Tsoi A.C., Back A.D. Face recognition: A convolutional neural-network approach. IEEE Transactions on Neural Networks. 1997;8(1): 98-113.

2. Li H., Lin Z., Shen X., Brandt J., Hua G. A neural network cascade for face detectiom. Proceedings of the IEEE conference on computer vision and pattern recognition. 2015:5325-5334.

3. Opitz M., Waltner G., Poier G., Possegger H., Bischof H. European conference on computer vision. Springer; Cham: 2016. Grid loss: Detecting occluded faces; pp. 386–402. October.

4. Li Y., Sun B., Wu T., Wang Y. European conference on computer vision. Springer; Cham: 2016. Face detection with end-to-end integration of a convnet and a 3d model; pp. 420-436. October.

5. Li C., Wang R., Li J., Fei L. Recent trends in intelligent computing, communication and devices. Springer; Singapore: 2020. Face detection based on YOLOv3; pp. 277-284.

6. Ben Krose and Pattrick van der Smagtt. An introtuction to Neural Networks. (1996). – 126.

7. Nguyen H. Fast object detection framework based on mobilenetv2 architecture and enhanced feature pyramid. Journal of Theoretical and Applied Information Technology. 2020; 98 (05).

8. Wang Z., Wang G., Huang B., Xiong Z., Hong Q., Wu H. Chen H. 2020. Masked face recognition dataset and application. arXiv preprint arXiv: 2003.09093.

9. Ojala T., Pietikainen M., Maenpaa T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2002;24(7):971–987.

10. Крижевский А., Суцкевер И., Джеффри Э. Хинтон Классификация Imagenet с глубокими сверточными нейронными сетями», «Достижения в системах обработки нейронной информации», 2012, с. 1097–1105.

11. Довідкова сторінка програми CuneiForm. Режим доступу: https://launchpad.net/cuneiform, вільний.

12. Інформацією про розпізнавач тексту FineReader. Режим доступу: https://www.abbyy.com/ru-ru/download/finereader/, вільний.

13. Ліндсей П., Нордман Д. Переробка інформації у людини. – М.: Мір, 1974. – 550 с.

14. Літюк В.И. Методичний посібник № 2231 частина 3 «Методи розрахунку і проектування цифрових багатопроцесорних пристроїв обробки радіосигналів», Таганрог, 1995, 48 с.

15. Мисюрёв А.В. Использование искусственных нейронных сетей для распознавания рукопечатных символов. В сб. "Интеллектуальные технологии ввода и обработки информации", М.: Эдиториал УРСС, 1998., 142 с.

16. Молинаро Энтони. SQL. Сборник рецептов / Энтони Молинаро. – O'Reilly, 2009. – 672 с.

17. Мэтью Д. Цейлер и Роб Фергус, «Визуализация и понимание сверточных сетей» в Computer Vision. 2014, стр. 818–833, Springer.

18. Мэтью Д. Цейлер, Грэм У. Тейлор и Роб Фергус, «Адаптивные деконволюционные сети для изучения функций среднего и высокого уровня», в Международной конференции IEEE по компьютерному зрению (ICCV), 2011, стр. 2018–2025.

19. Нейронні мережі та основні функції. Режим доступу: https://clck.ru/GLKYq, вільний.

20. Оптичне розпізнавання символів (OCR), аналіз. Режим доступу https://www.nicomsoft.com/optical-character-recognition-ocr-how-it-works/, вільний.

21. Путятин Е.П., Аверин С.И. Обработка изображений. – М: Машиностроение, 2010. – 320 с

22. Структура програмного стеку бібліотеки Tensorflow. Режим доступу: https://www.tensorflow.org/images/tensorflow_programming_environment.png, вільний.

23. Шар відкидання в нейронних мережах. Режим доступу: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/ dropout_layer.html, вільний.

24. Штучний інтелект. Системи та експертні системи. Кн. 1 / Під ред. Э.В.Попова. – М.: Радіо та зв'язок, 1990. – 461 с.

25. Щепин Е.В., Непомнящий Г.М. К топологическому подходу в анализе изображений. Геометрия, топология и приложения (Межвузовский сборник научных трудов). Москва, Мин. высшего и средн. спец. образ. РСФСР, Московский институт приборостроения, 1990., 315 с.

26. ДСТУ 3008-95 Документація. Звіти у сфері науки. Структура і правила оформлення.

27. НД ТЗІ 1.1-003-99. Термінологія у області захисту інформації в комп’ютерних системах від несанкціонованого доступу. // Департамент спеціальних систем і захисту інформації Служби безпеки України. – Київ, 1999.

4. Directory

4.1. Index

Part 2.1

δ – the threshold value.

Part 2.1.1

X is the input, a single feature that we provide to our model to calculate the forecast.

B_1 is the calculated slope of our logistic regression - B_1 tells us how much Log_Odds change when X changes. Note that B_1 lives on the turquoise line that connects the X input to the blue neuron in the hidden layer 1.

B_0 is a bias - very similar to regression interception. The key difference is that in neural networks, each neuron has its own offset period (whereas in regression, the model has a special interception period).

Part 2.1.2.

[W] – a matrix of weights n on m (connections between the previous layer and the current layer),

[X] – a matrix of m by 1 or initial inputs, or activations from the previous layer, [Bias] is your n by 1 matrix of neural prejudices,

[Z] – n per 1 matrix of intermediate outputs. In the previous equation I follow the Python notation,

@ - designation of multiplication of matrices.*Part 3.4:*

Tp – True positive,

Tn – True negative,

Fp – False positive,

Fn – False negative.

4.2. Glossary

Artificial neural networks (ANN) are a hightech based on studies of the brain and nervous extensions as depicted in Fig. 1. These solutions emulate a biological neural network but they use a minimized set of concepts from biological neural systems. Specifically, ANN models simulate the practical activity of the brain and nervous system.

Machine vision (MV) – is the technology and methods used to distinguish imaging-based automatic inspection and tactics for such applications as automatic analysis, process control, and robot guidance, usually in companies. Machine vision

refers to many technologies, applications and hardware products, integrated systems, actions, fragments and expertise. Machine vision as a systems engineering discipline can be considered distinct from machine vision, a form of tech science. It attempts to integrate existing applications in new ways and apply them to solve real world problems. The term is the prevalent one for these methods in industrial automation environments but is also used for these functions in other environment automotive guidance.

4.3. Acronym List

ANN	–	*artificial neural networks*
GPU	–	*graphics processing unit*
HTTP	–	*hyper text transfer protocol*
ISO	–	*international organization for standardization*
NIST	–	*National Institute of Standards and Technology*
OpenCV	–	*Open Source Computer Vision Library*
OCR	–	*Optical Character Recognition*
PNG	–	*Portable Network Graphics*
HMM	–	Hidden Markov Models
TVS	–	Technical Vision System
CAF	–	Conditional Arbitrary Fields

5. Sample Figures & Tables

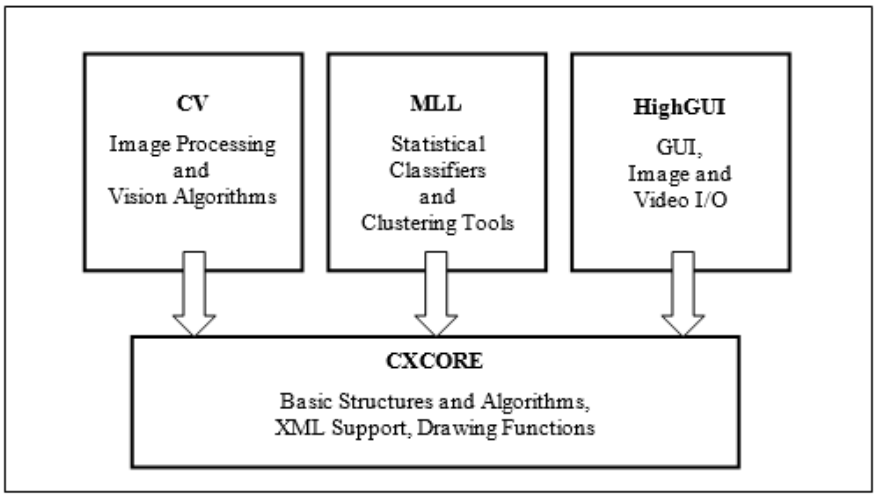


Рис. 1.1. Структура бібліотеки *OpenCV*

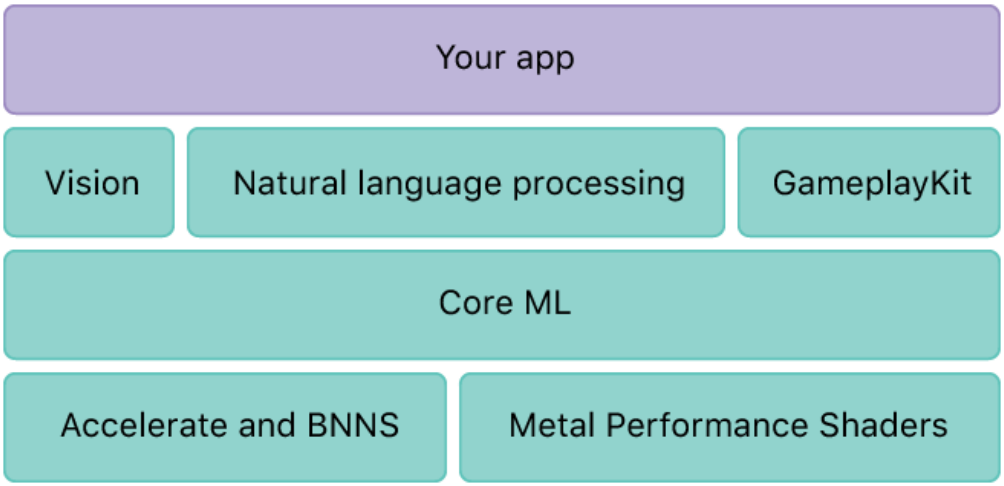


Рис 1.2. Структура *CoreML*

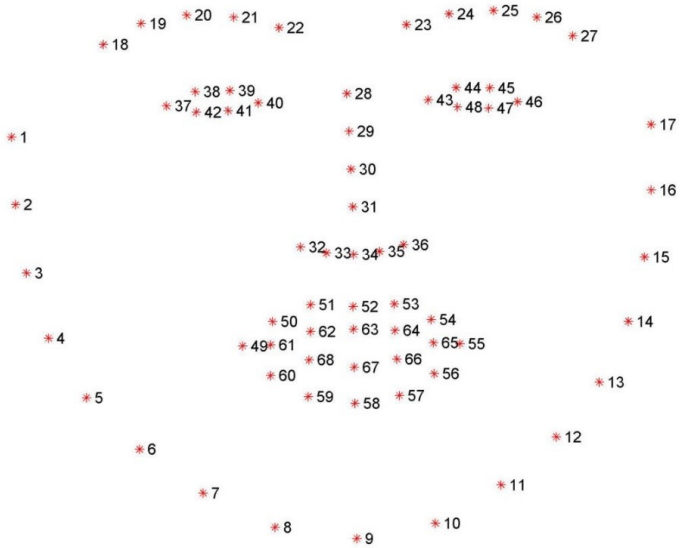


Рис. 1.3. Приклад визначення *landmarks* в *DLIB*

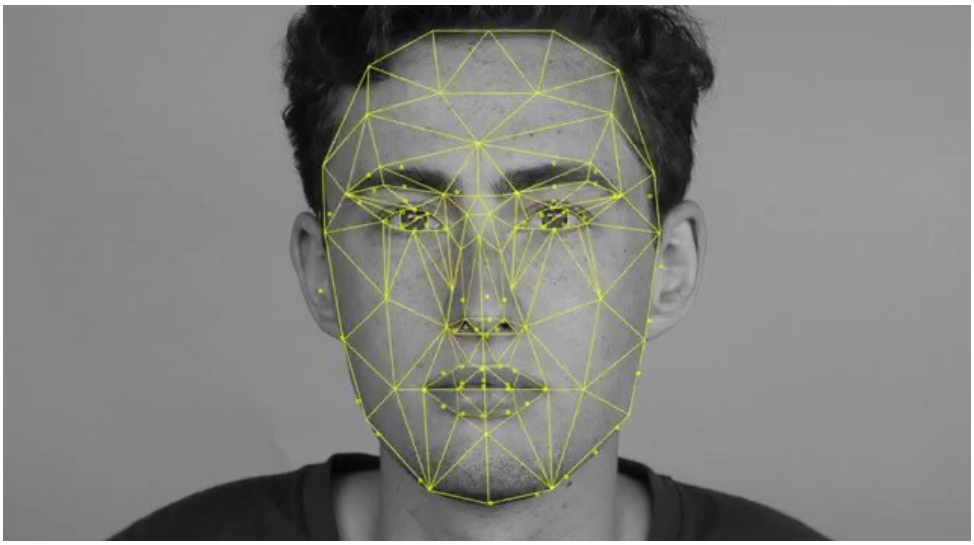


Рис. 1.4. Маска, що візуалізує алгоритм тріангуляції Делоне

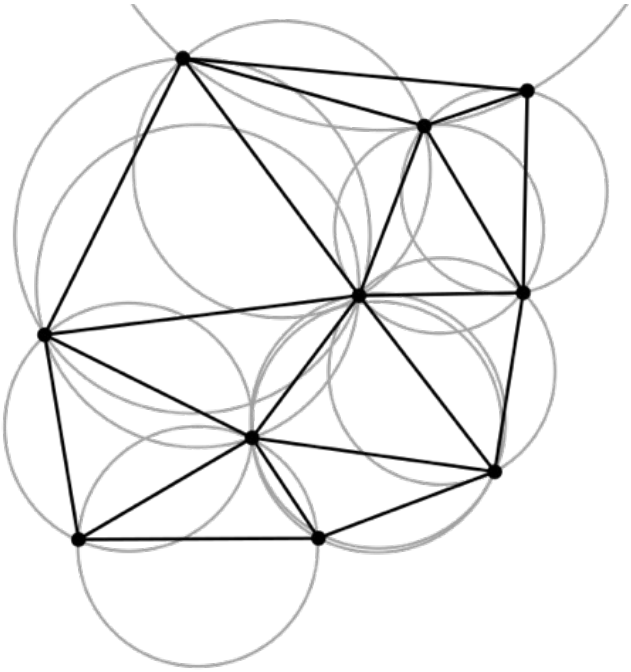


Рис. 1.5. Приклад тріангуляції Делоне

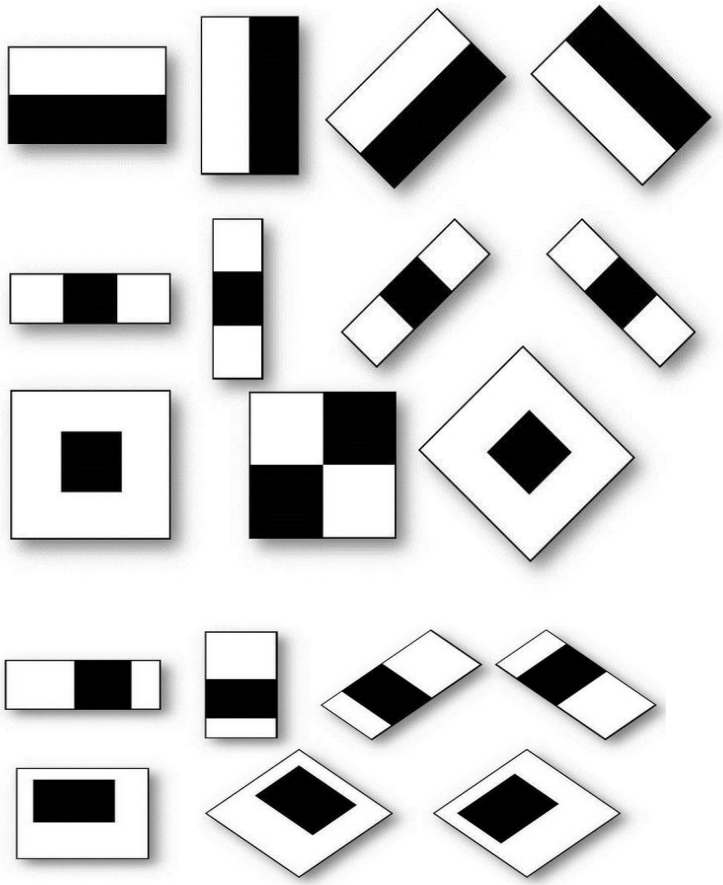
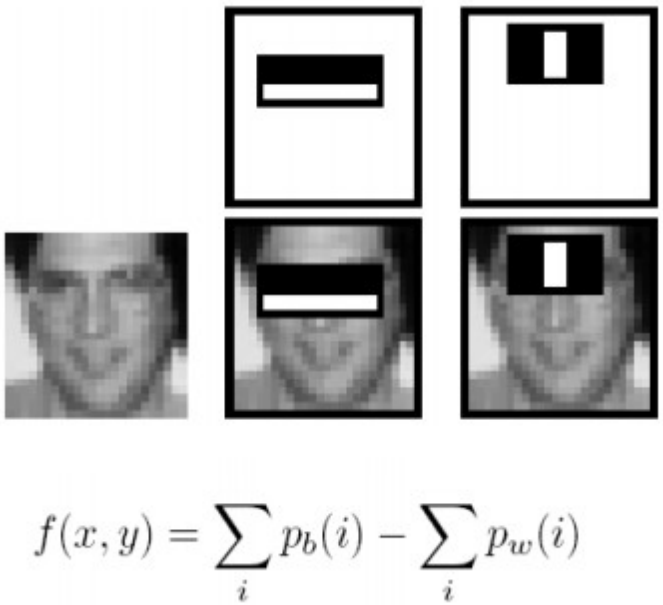


Рис. 1.6. Примітиви Хаара



$$f(x,y) = \sum_i p_b(i) - \sum_i p_w(i)$$

Рис. 1.7. Приклад пошуку ознак обличчя за допомогою примітивів Хаара

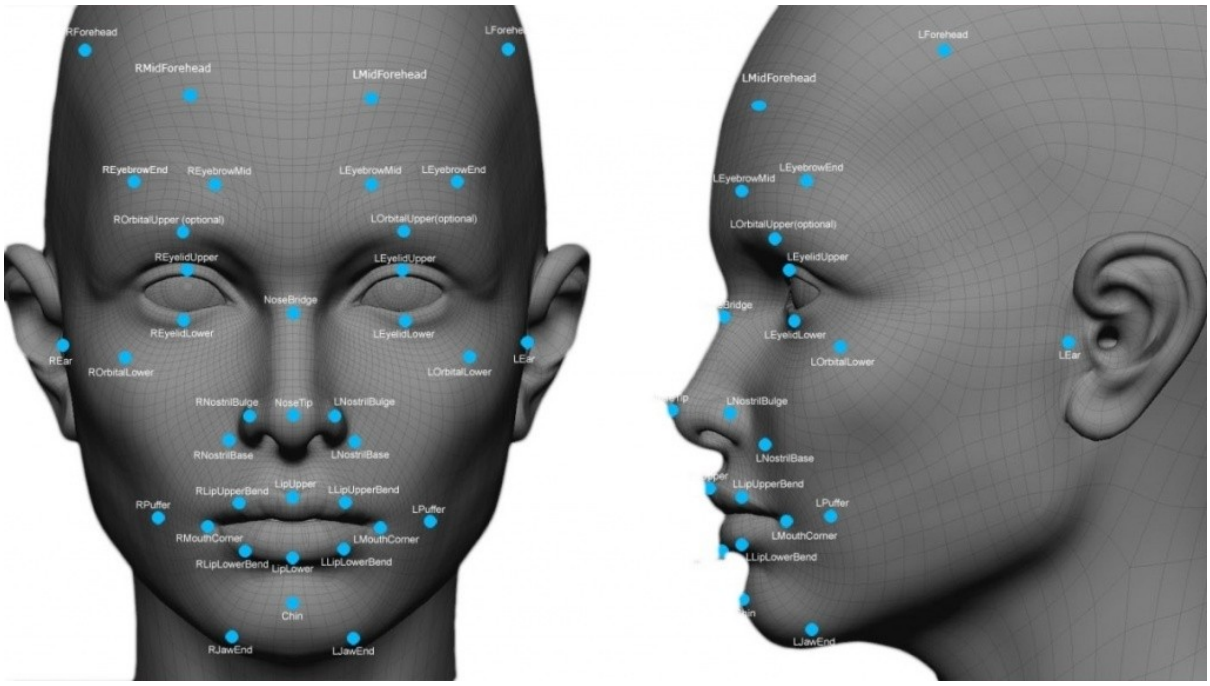


Рис. 1.8. Приклад антропометричних точок

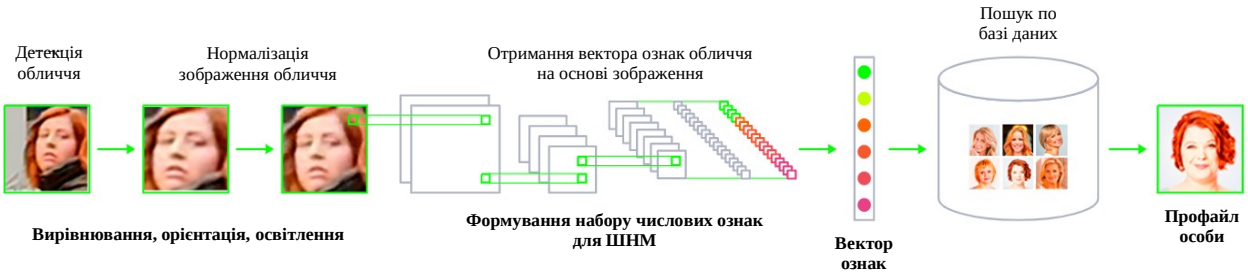


Рис. 1.9. Етапи визначення особи на основі зображення обличчя

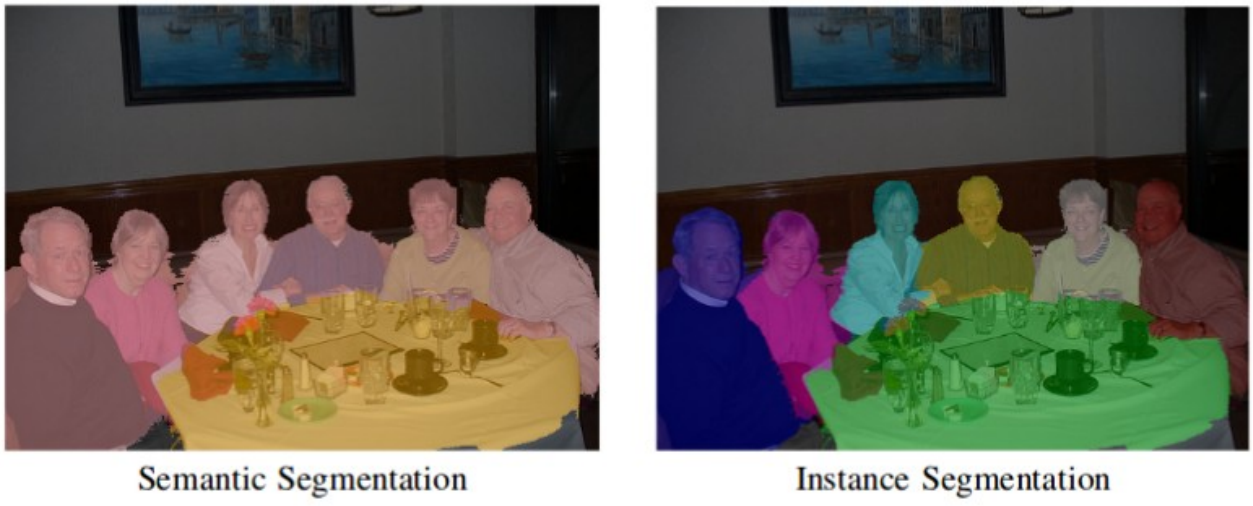


Рис. 1.10. Порівняння інстанс- та семантичної сегментації

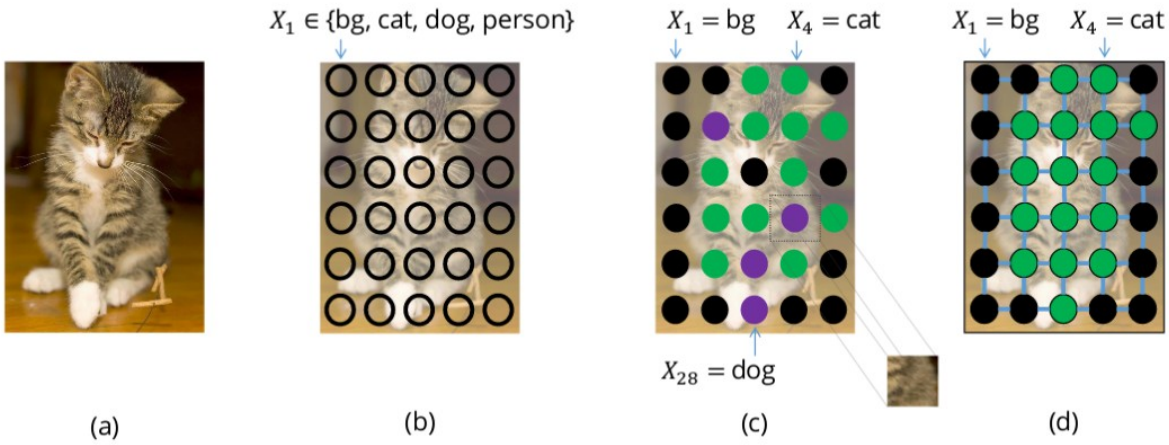


Рис. 1.11. Приклад зашумленої сегментації: *a*) зображення, *b*) сегментація, *c*) пікселі з мітками собак змішані з мітками котів, *d*) більш природна сегментація

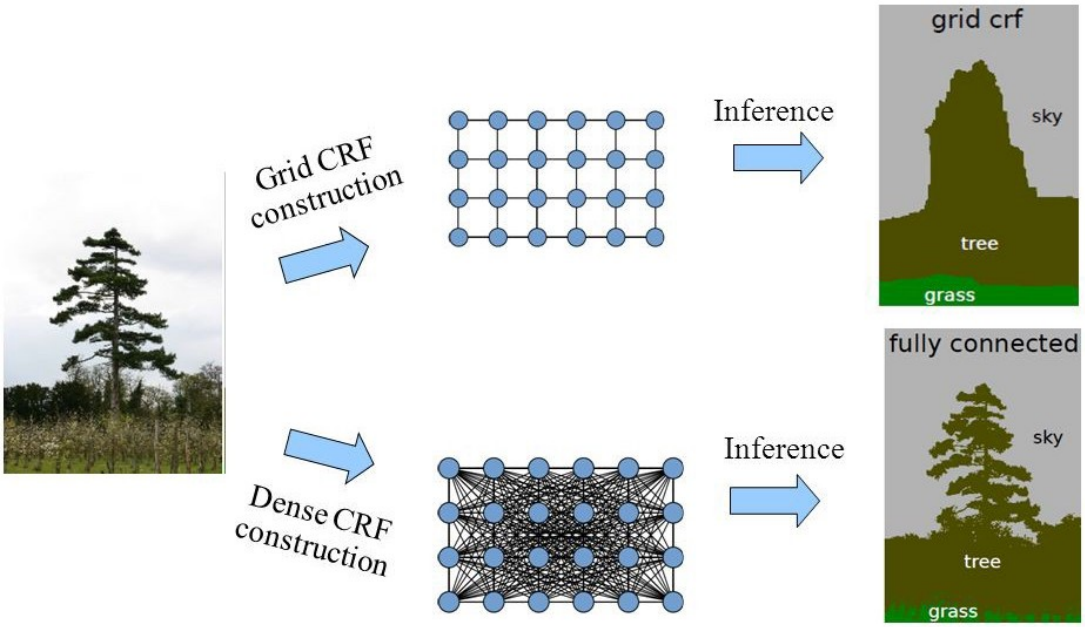


Рис. 1.12. Порівняння сіткових (*Grid CRF*) та щільних (*Dense CRF*) УДП

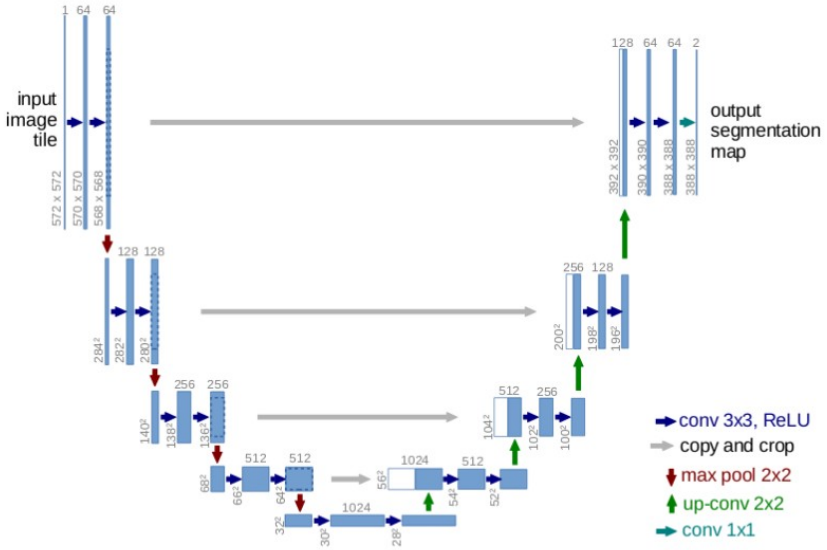


Рис. 1.13. Схема *U-Net* мережі

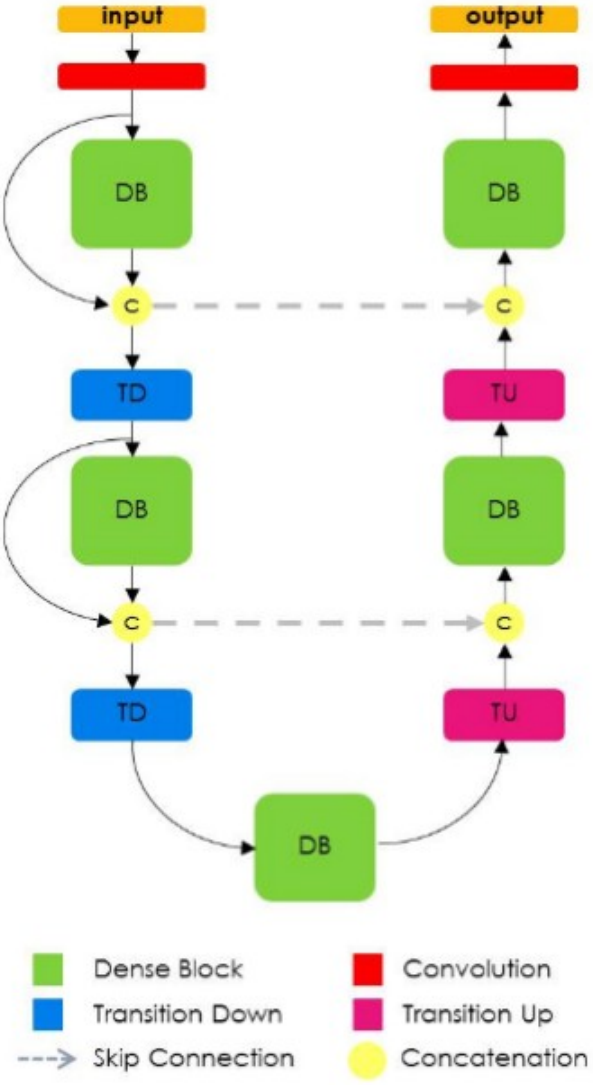


Рис. 1.14. Схема мережі Тіраміс

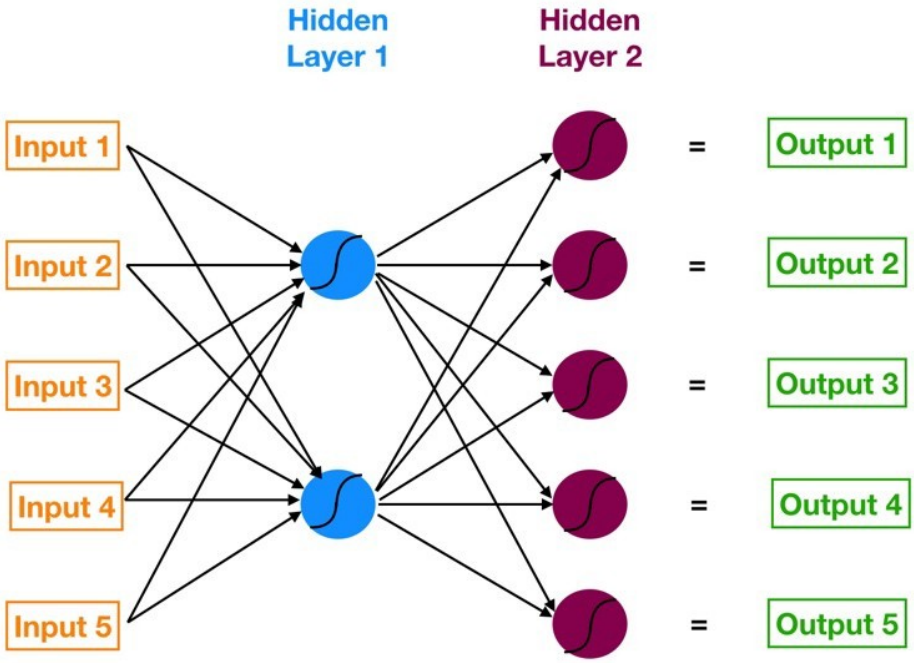
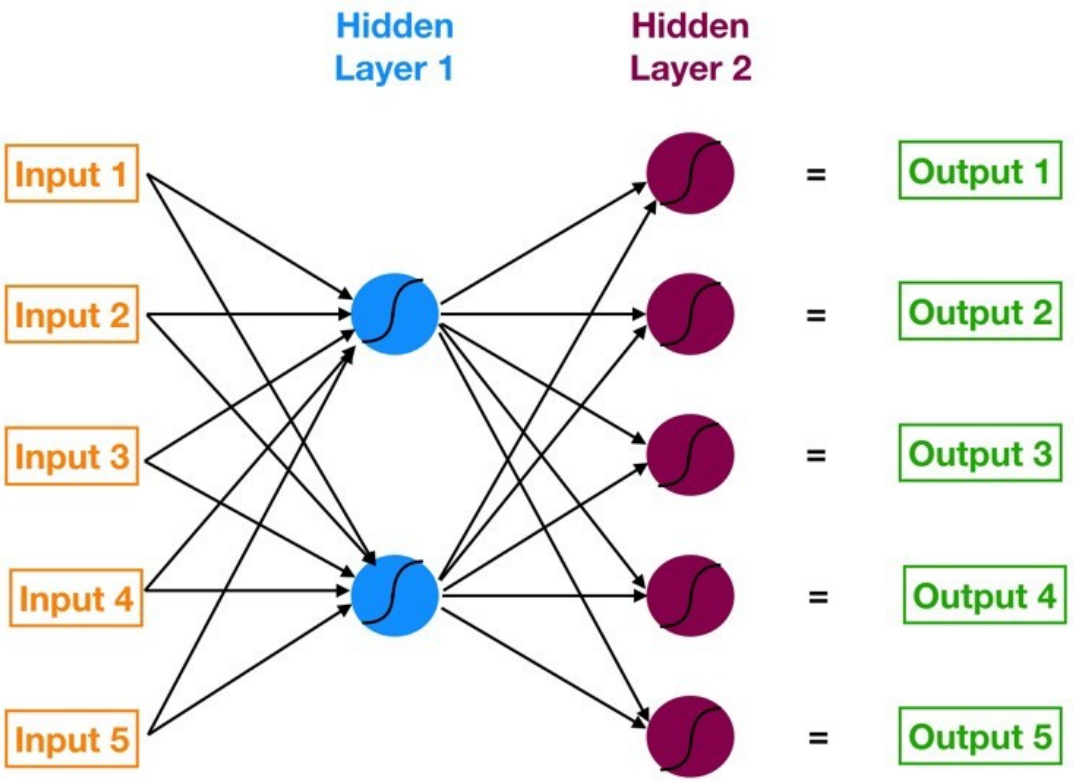


Рис. 2.1. Нейромережа з двома прихованими шарами



\

Рис. 2.2. Ускладнена нейронна мережа

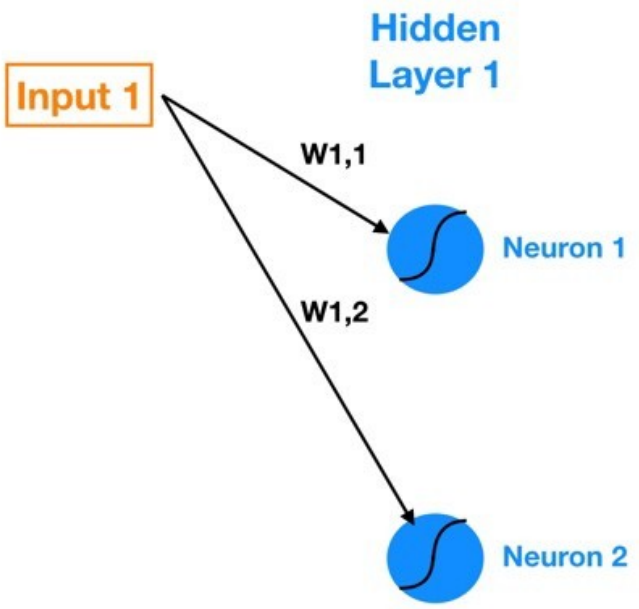


Рис. 2.4. Зв'язки між входом 1 і прихованим шаром 1

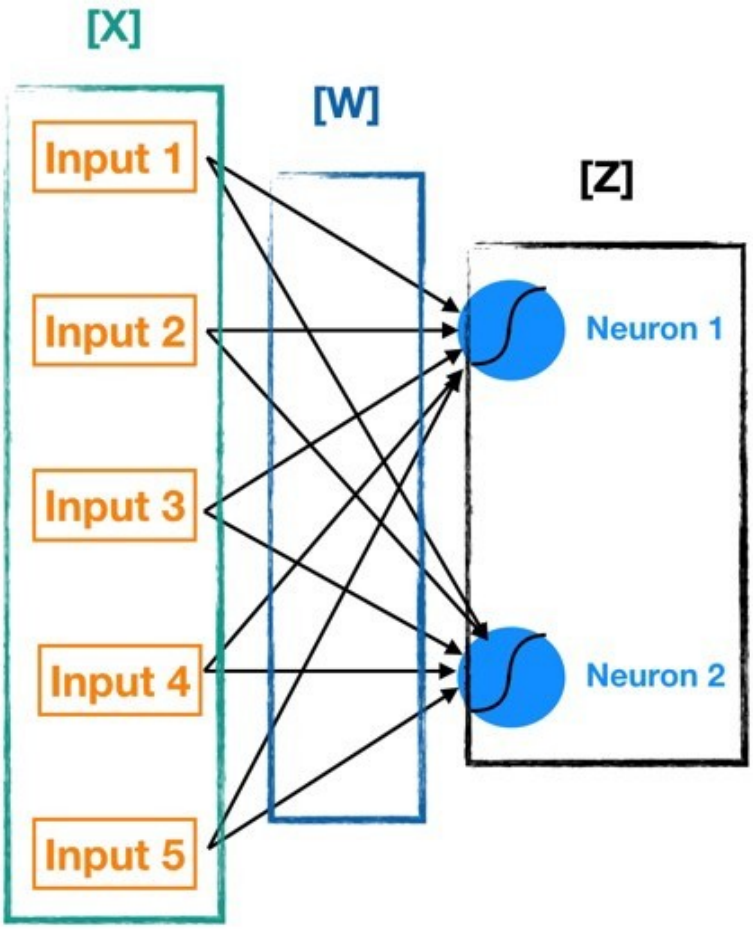


Рис. 2.5. Візуалізація [*W*], [*X*] та [*Z*]

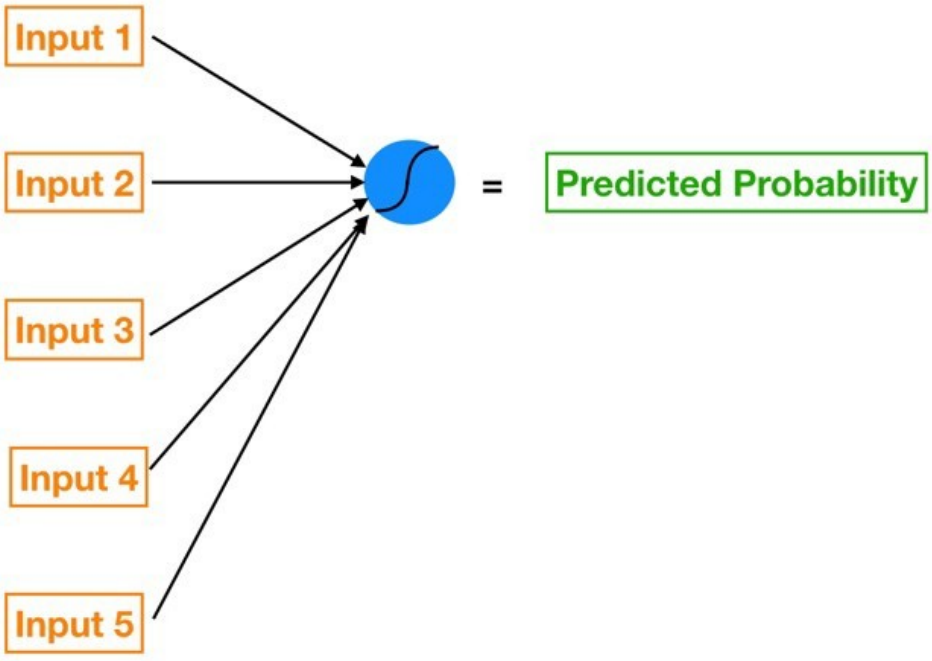


Рис. 2.6. Представлення логістичної регресії

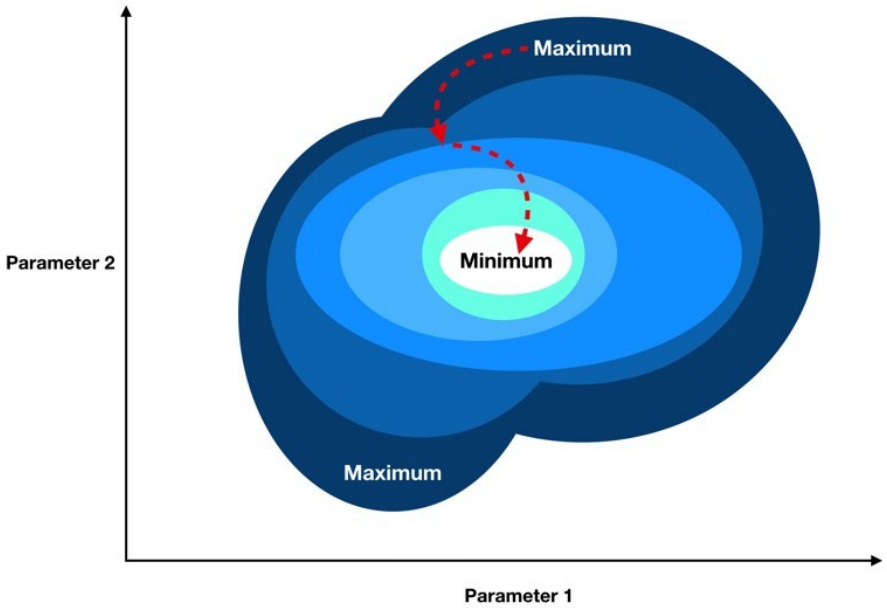


Рис. 2.7. Ілюстрація градієнтного спуску

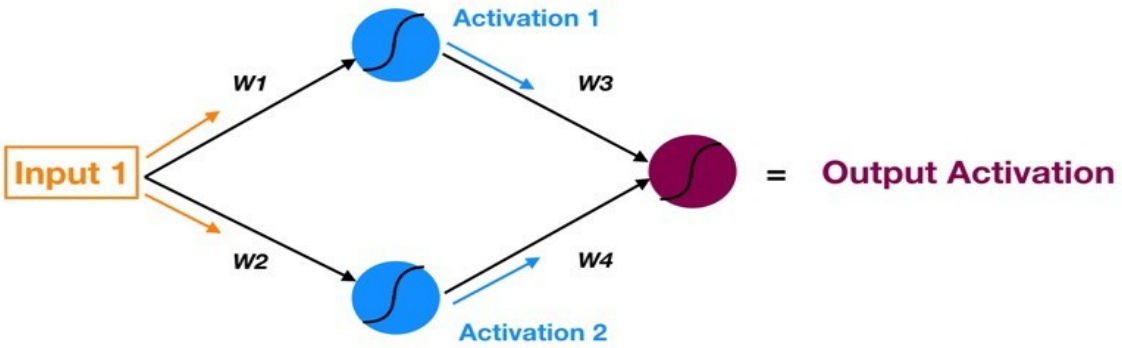


Рис. 2.8. Поширення вперед у нейронній мережі

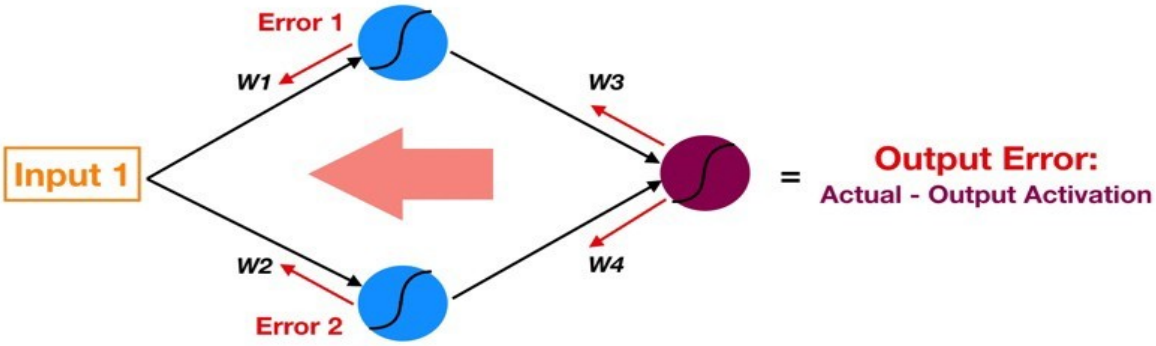


Рис. 2.9. Зворотне розмноження в нейронній мережі

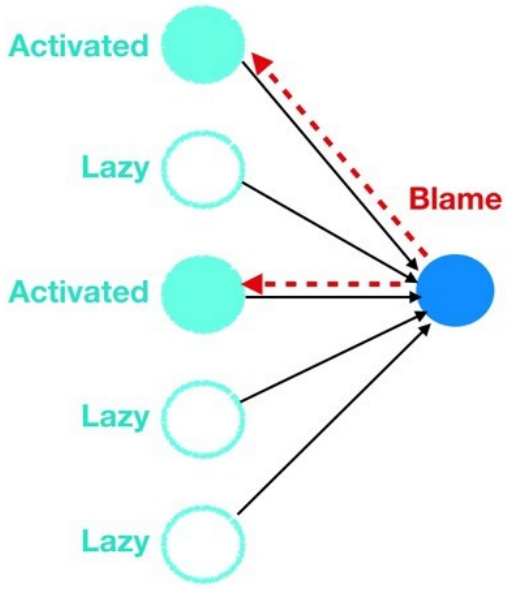


Рис. 2.10. Нейрони звинувачують найбільш активні нейрони вище по течії

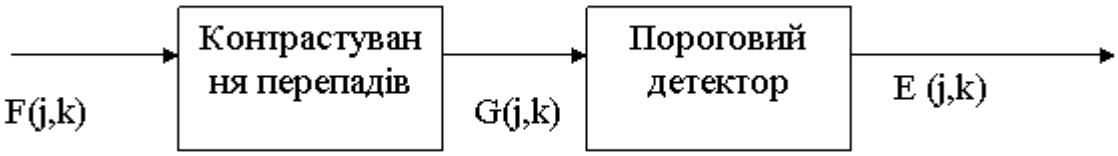


Рис. 2.11. Порогова система виявлення перепадів

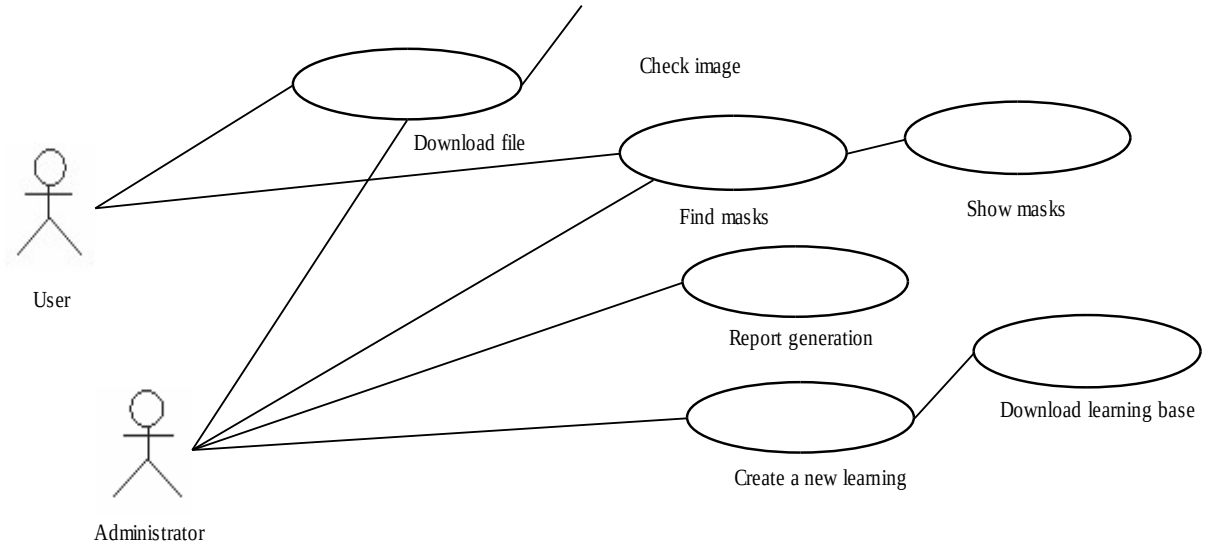


Рис. 3.1. Діаграма прецедентів використання системи

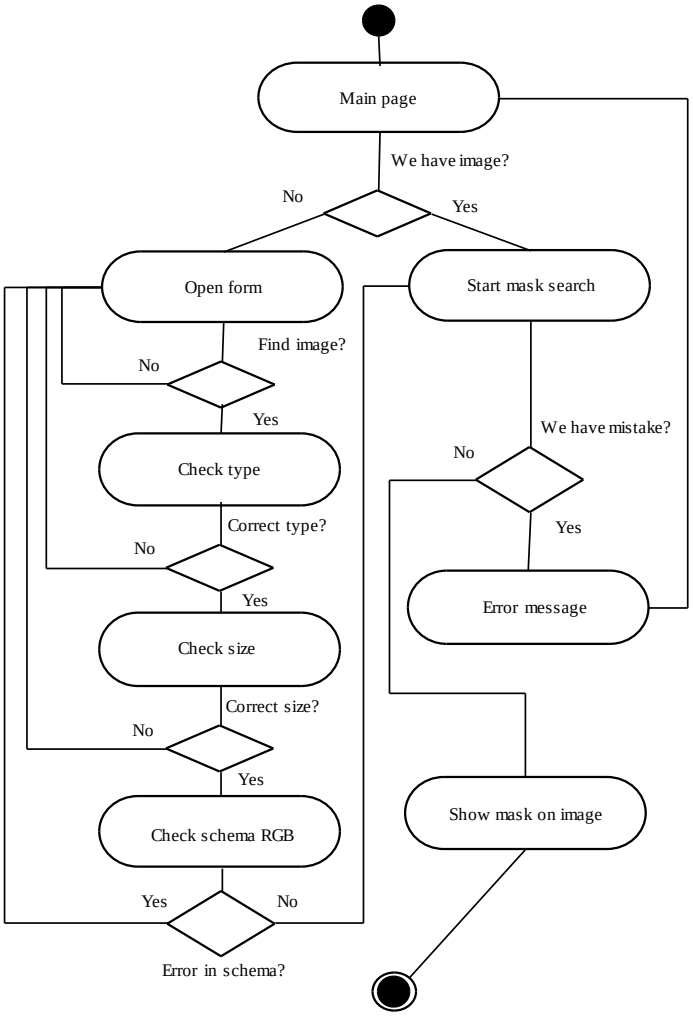


Рис. 3.2. Діаграма послідовності дій для запису даних в системі

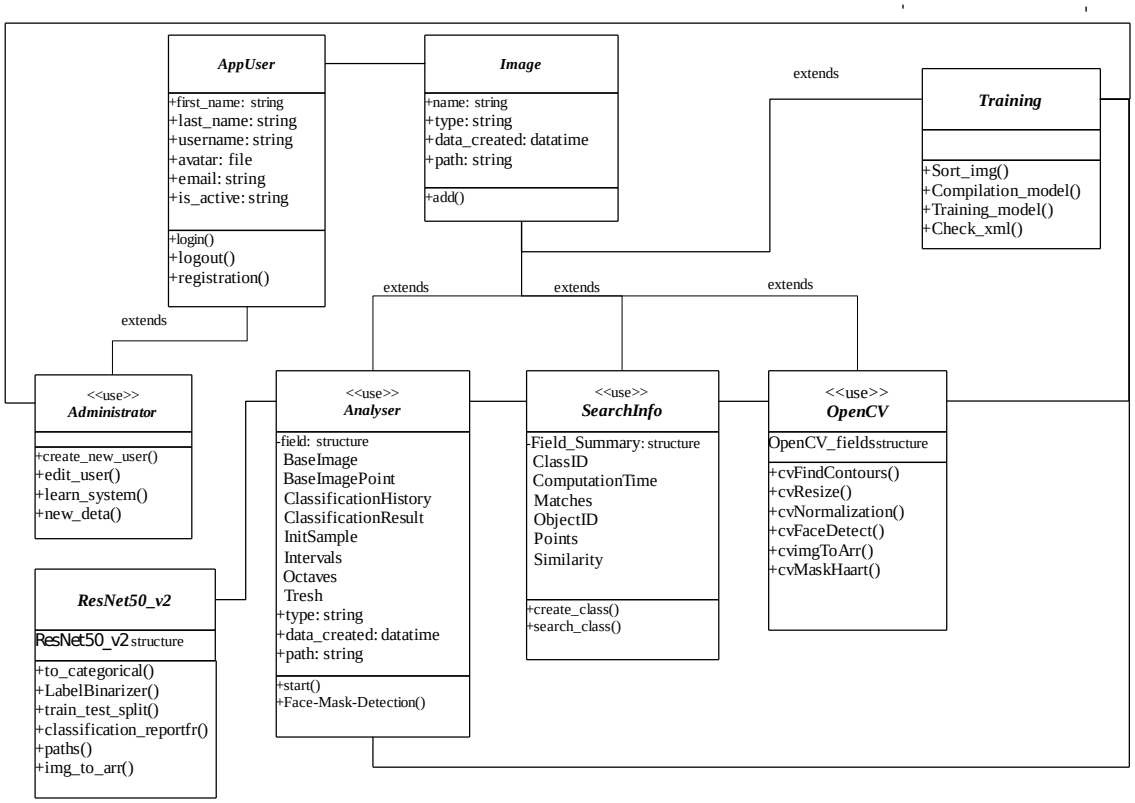


Рис. 3.3. Діаграма класів системи обліку стану здоров’я пілотів авіакомпанії

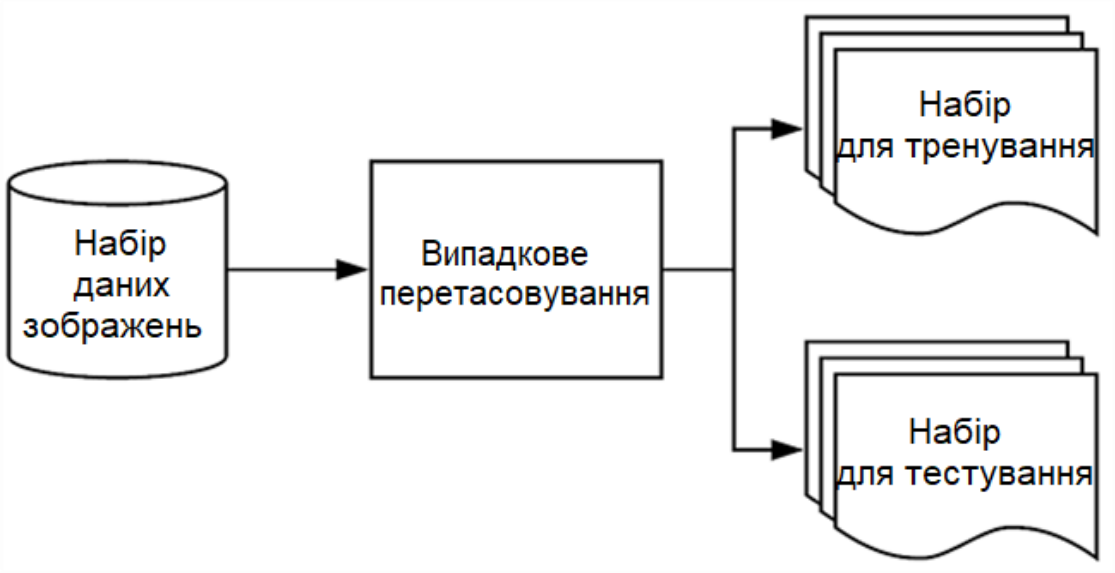


Рис. 3.4. Схема формування наборів даних для тренування та тестування

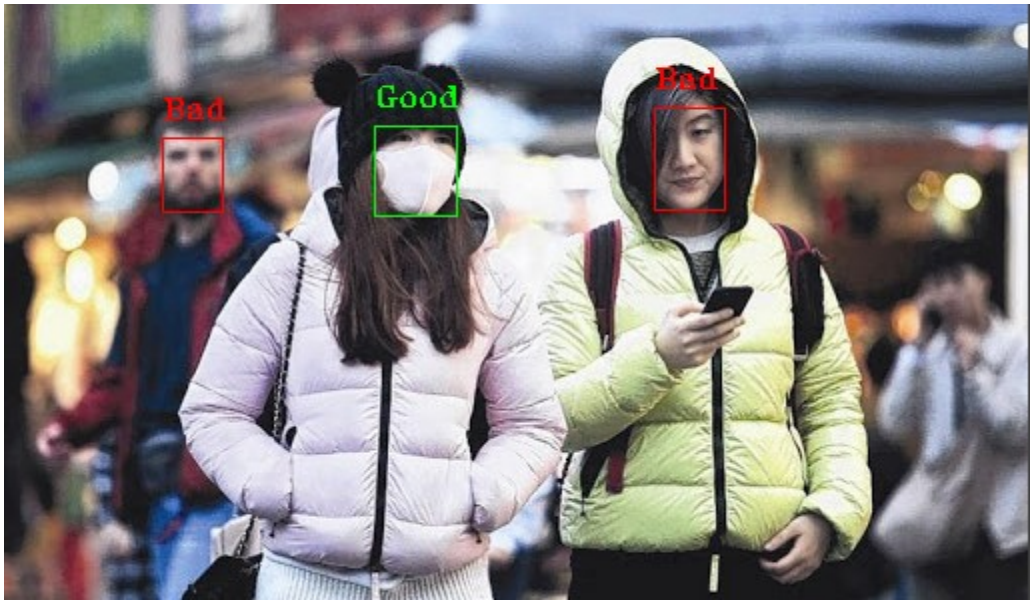


Рис. 3.5. Результат роботи скрипта *check_xml.py*

```
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.794565), count: 16, class_loss = 1.082497, iou_loss = 0.577214, total_loss = 1.659710
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.006939, iou_loss = 0.000000, total_loss = 0.006939
 total_bbox = 266411, rewritten_bbox = 0.007507 %
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.807932), count: 3, class_loss = 0.401827, iou_loss = 0.150494, total_loss = 0.552321
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.764739), count: 9, class_loss = 0.908124, iou_loss = 0.623584, total_loss = 1.531708
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000466, iou_loss = 0.000000, total_loss = 0.000466
 total_bbox = 266423, rewritten_bbox = 0.007507 %

 (next mAP calculation at 7000 iterations)
 Last accuracy mAP@0.5 = 86.81 %, best = 87.16 %
 7000: 0.785188, 0.777734 avg loss, 0.000001 rate, 13.053253 seconds, 448000 images, 0.245283 hours left
Resizing to initial size: 704 x 704  try to allocate additional workspace_size = 142.74 MB
 CUDA allocate done!

 calculation mAP (mean average precision)...
 Detection layer: 82 - type = 28
 Detection layer: 94 - type = 28
 Detection layer: 106 - type = 28
88
 detections_count = 764, unique_truth_count = 391
class_id = 0, name = Good, ap = 92.08%           (TP = 294, FP = 21)
class_id = 1, name = Bad, ap = 81.03%            (TP = 51, FP = 6)

 for conf_thresh = 0.25, precision = 0.93, recall = 0.88, F1-score = 0.90
 for conf_thresh = 0.25, TP = 345, FP = 27, FN = 46, average IoU = 72.63 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.865577, or 86.56 %
Total Detection Time: 3 Seconds
```

Рис. 3.6. Логування процесу обробки даних

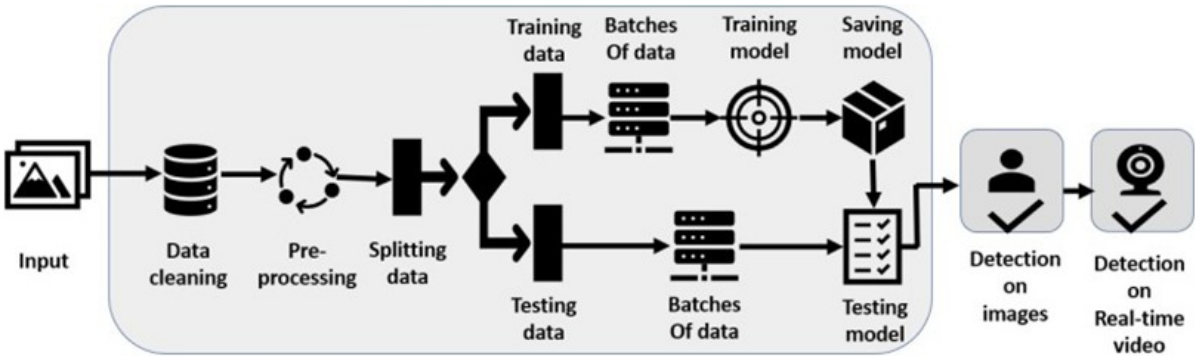


Рис. 3.7. Схема роботи моделі

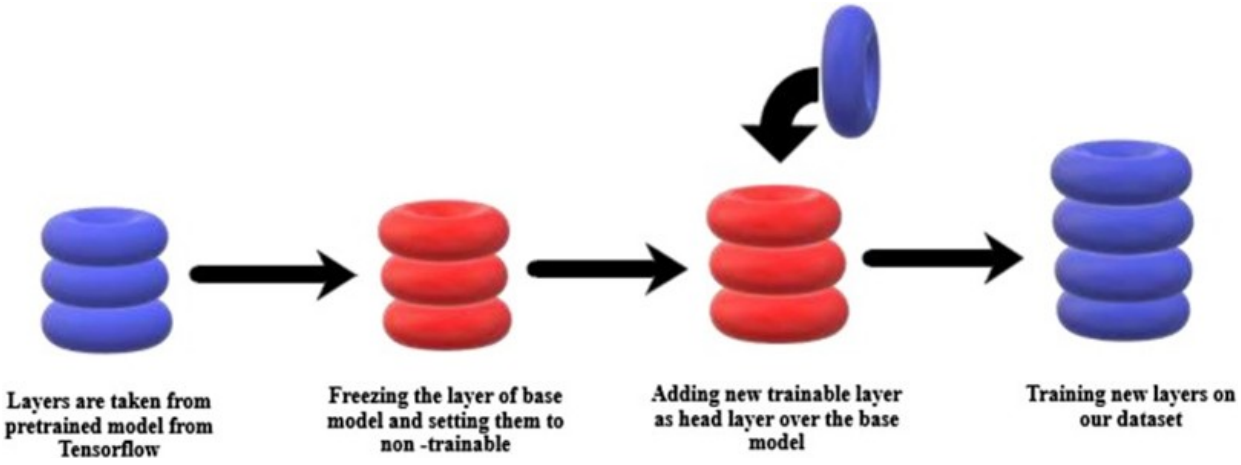


Рис. 3.8. Конвеєр використання попередньо підготовленої моделі

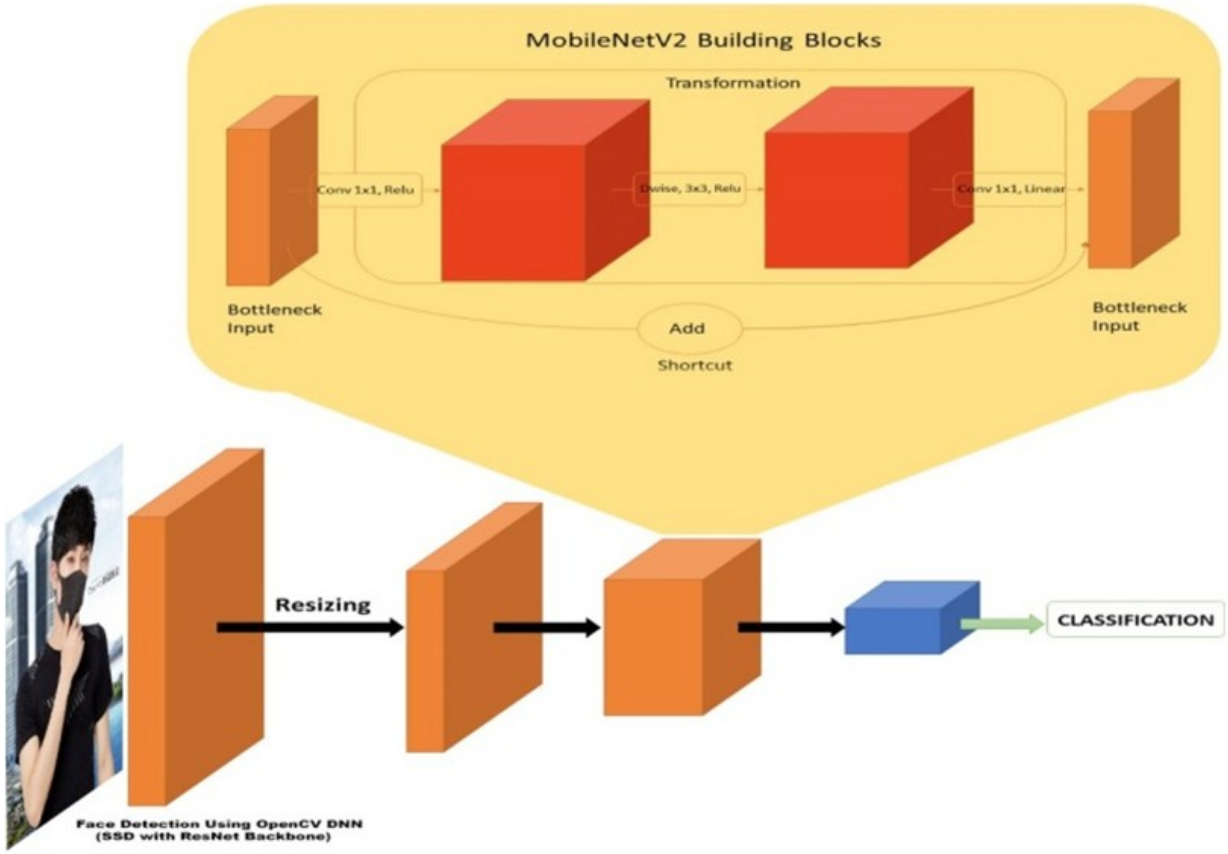


Рис. 3.9. Архітектура *MobileNetV*2

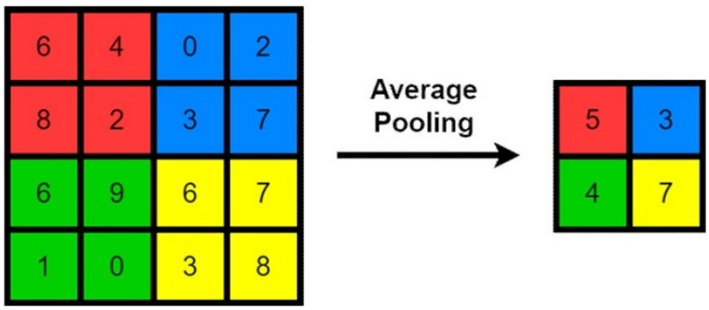


Рис. 3.10. Операція об’єднання середнього значення

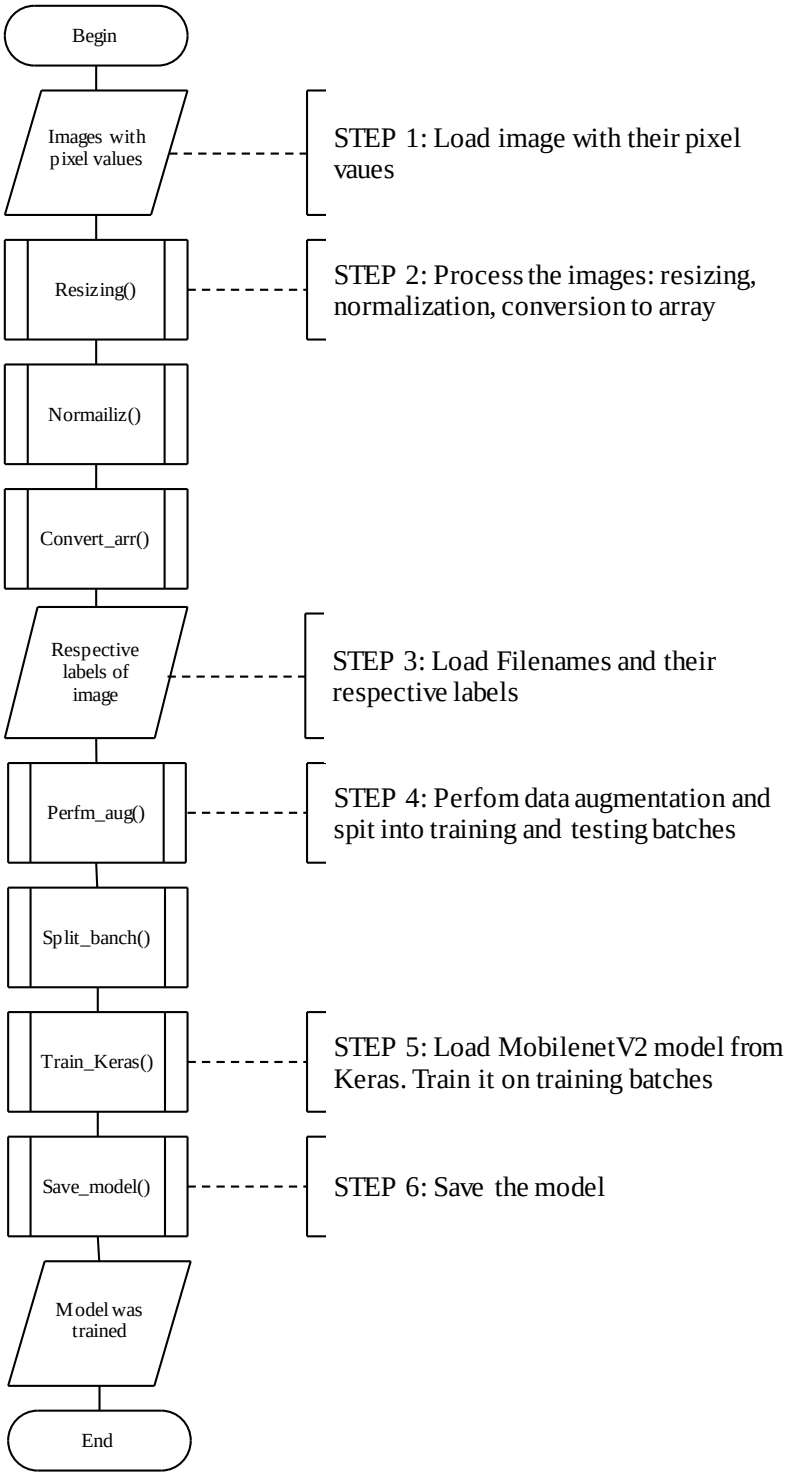


Рис. 3.11. Схема алгоритму попередньої обробки і тренування моделі

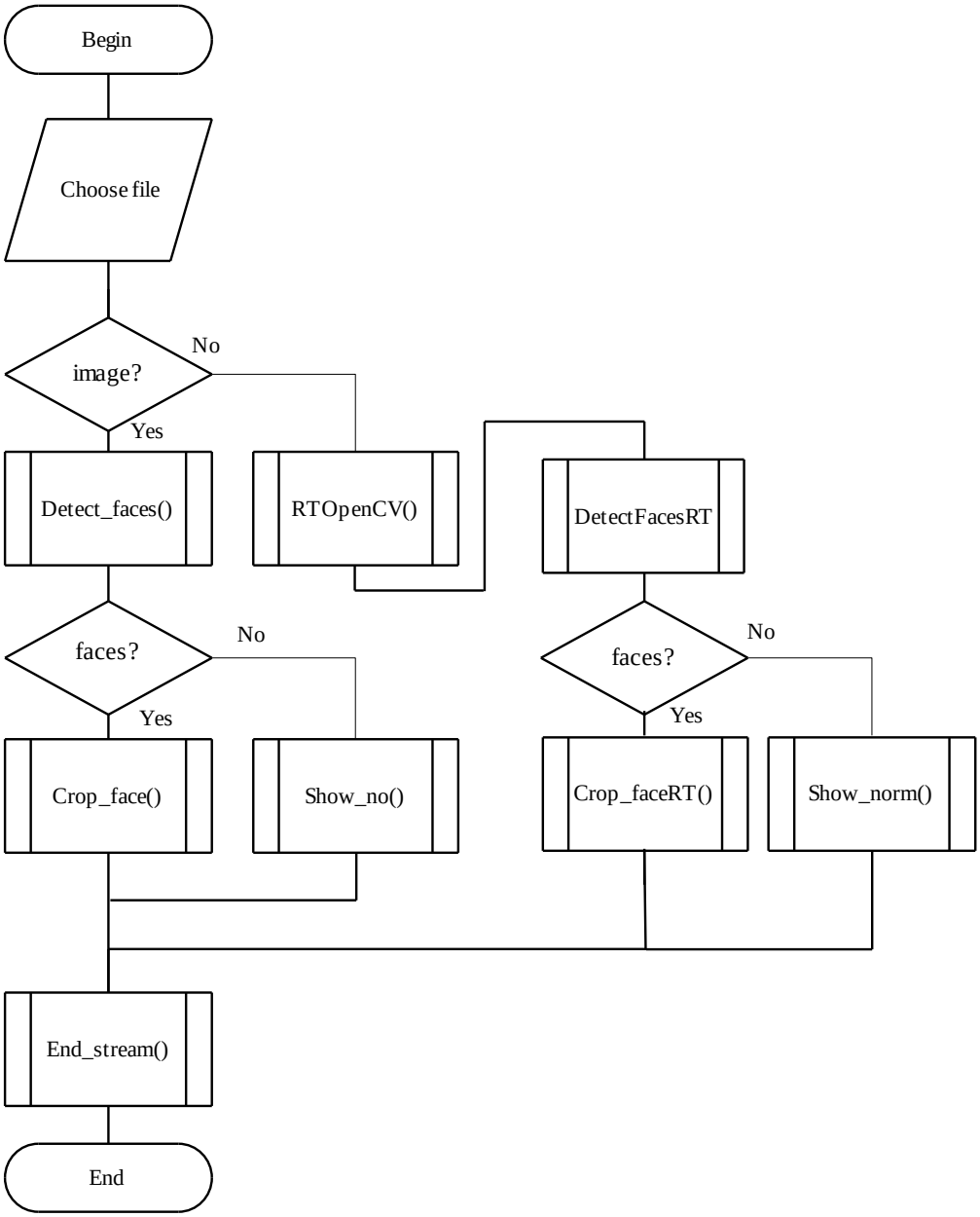


Рис. 3.12. Схема алгоритму визначення медичних масок на обличчях у зображеннях і у відеофайлах

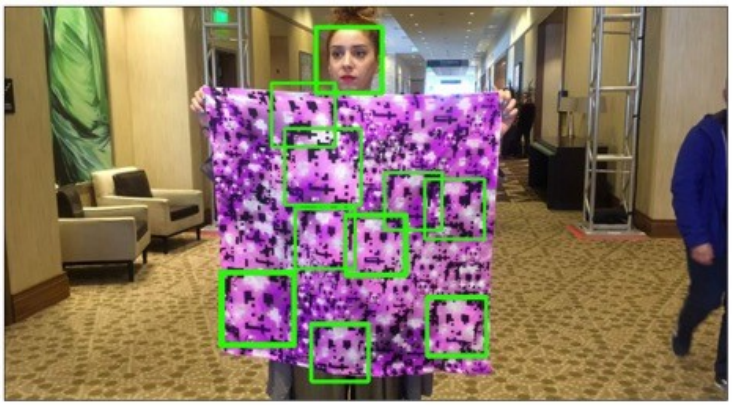


Рис. 3.13. Приклад зображення з маскуючими елементами

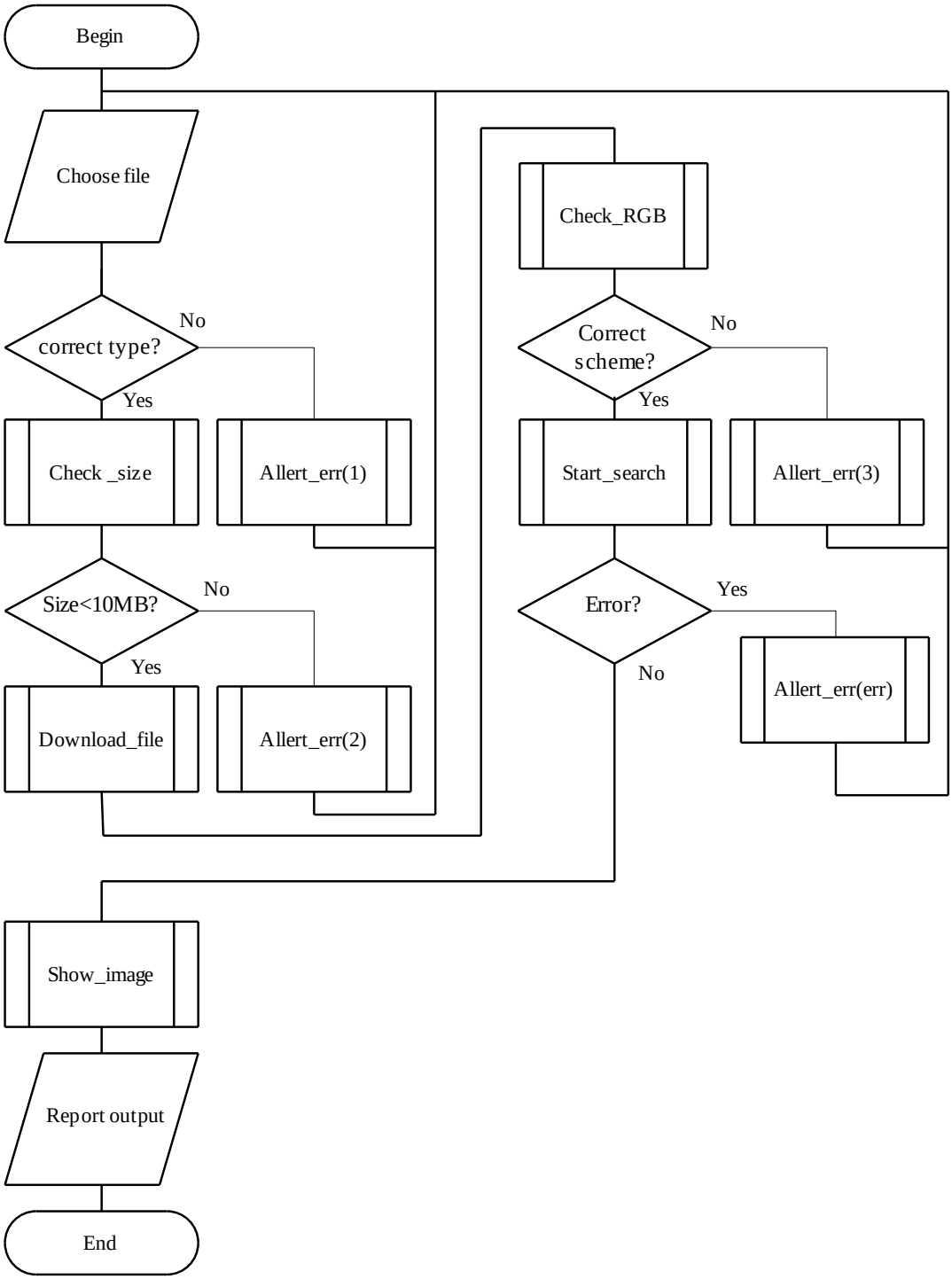


Рис. 3.14. Схема алгоритму роботи основного вікна системи визначення медичних масок на обличчях у зображеннях

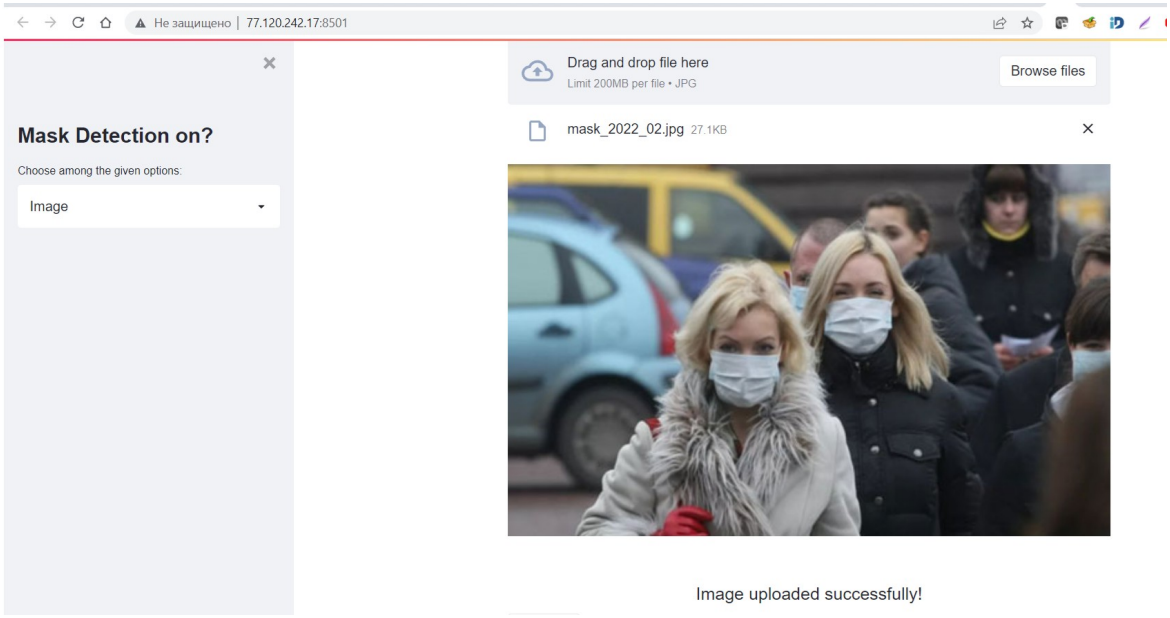


Рис. 3.15. Вікно системи з результатом завантаження зображення

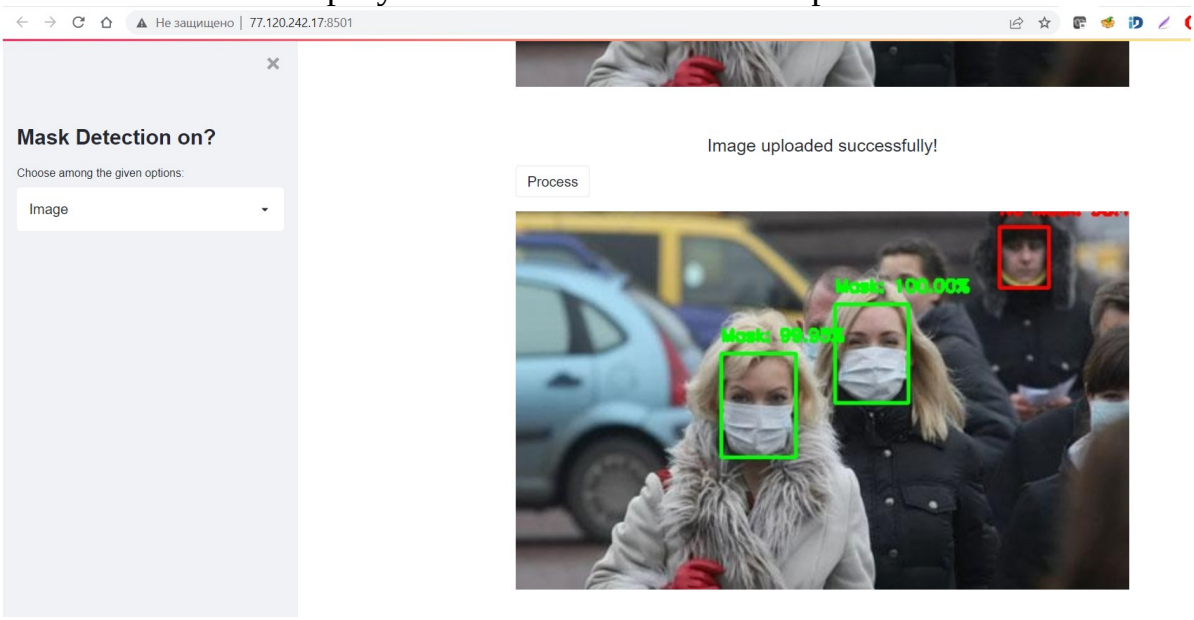


Рис. 3.16. Вікно системи з позначками про наявність масок на обличчах

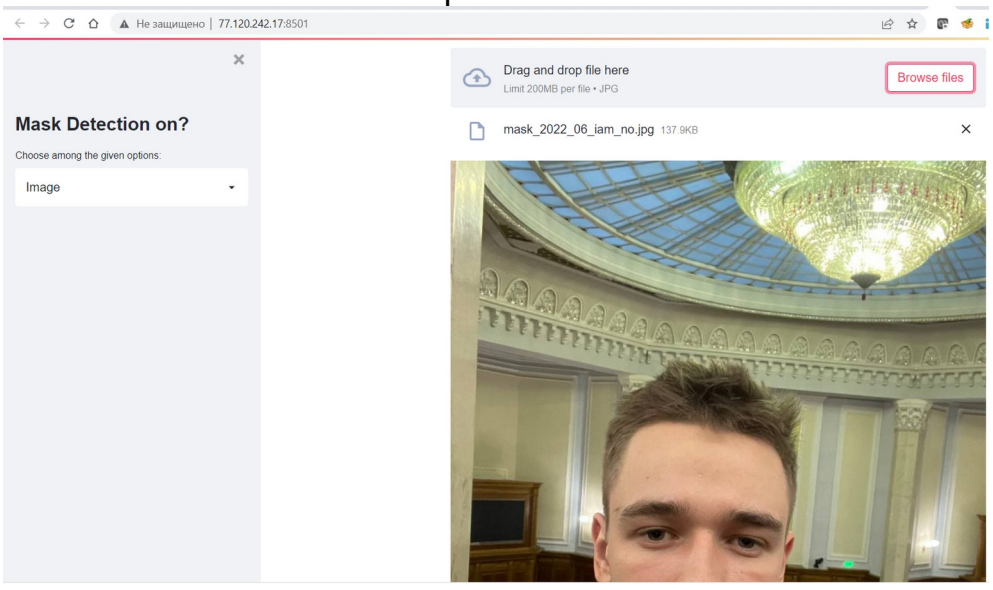


Рис. 3.17. Вікно системи з вертикально орієнтаціє зображення

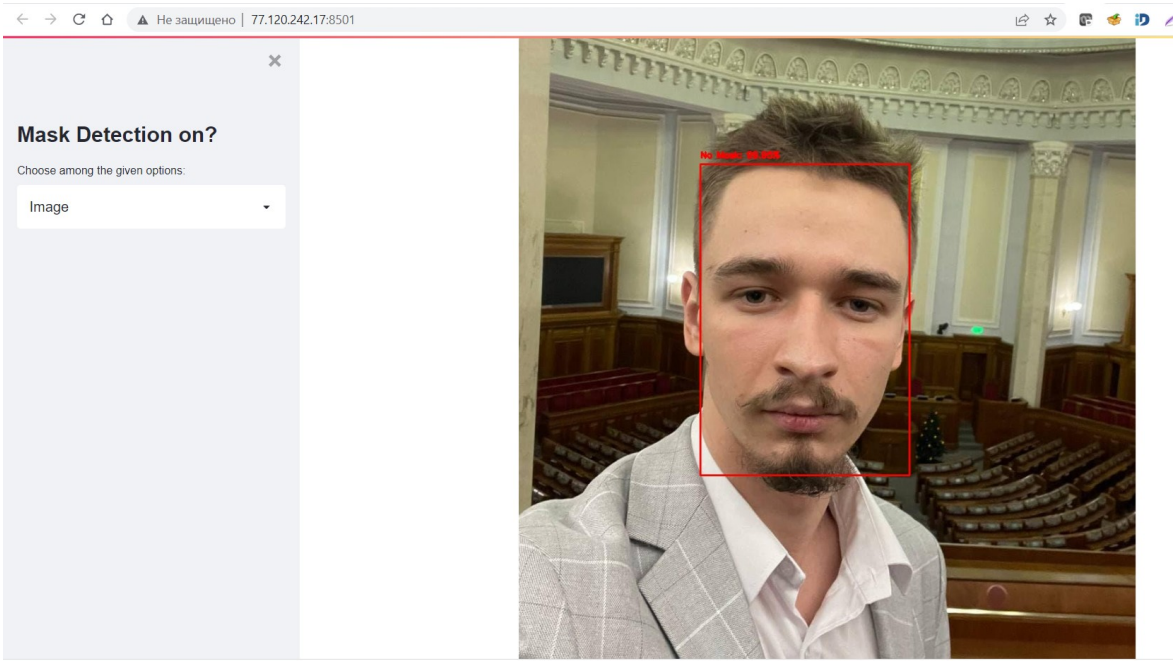


Рис. 3.18. Вікно системи з знайденним обличчям без маски

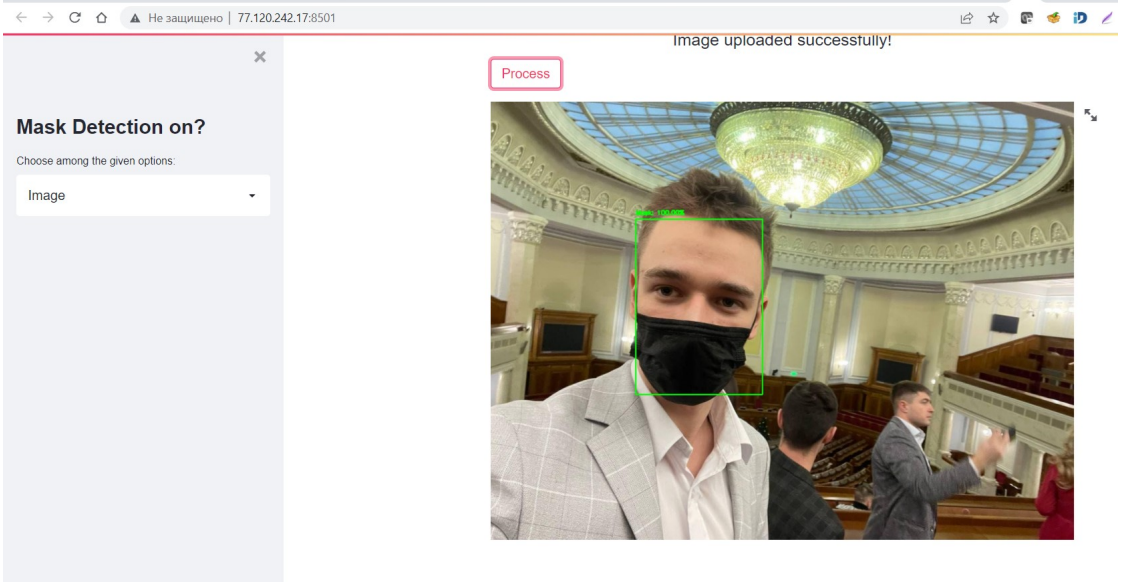


Рис. 3.19. Вікно системи з помилкою у відокремленні обличь

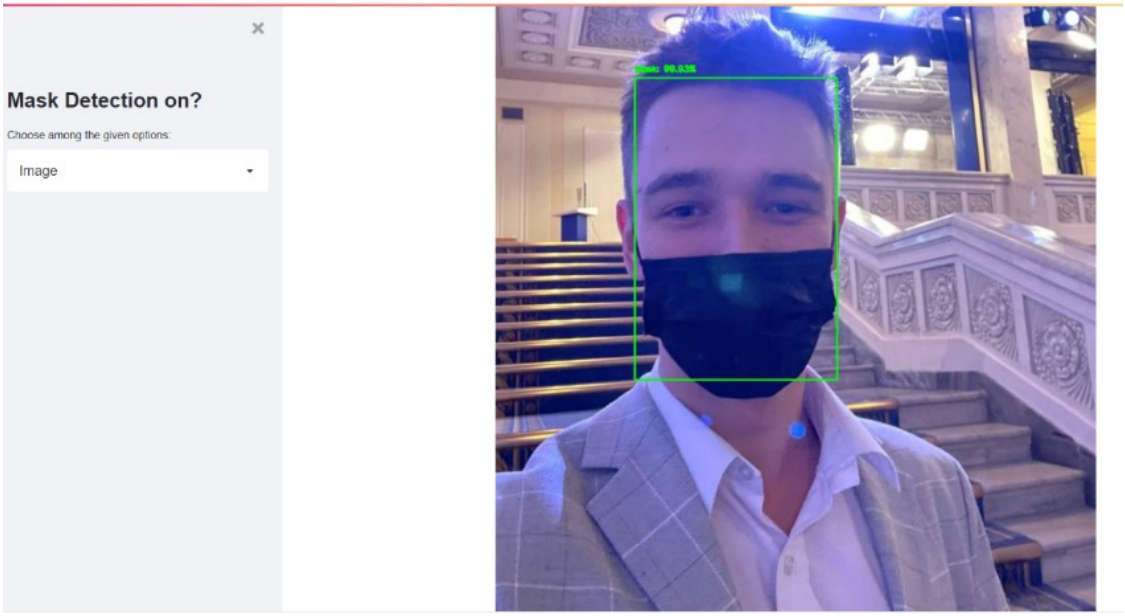


Рис. 3.20. Приклад роботи системи з нечіткими зображеннями

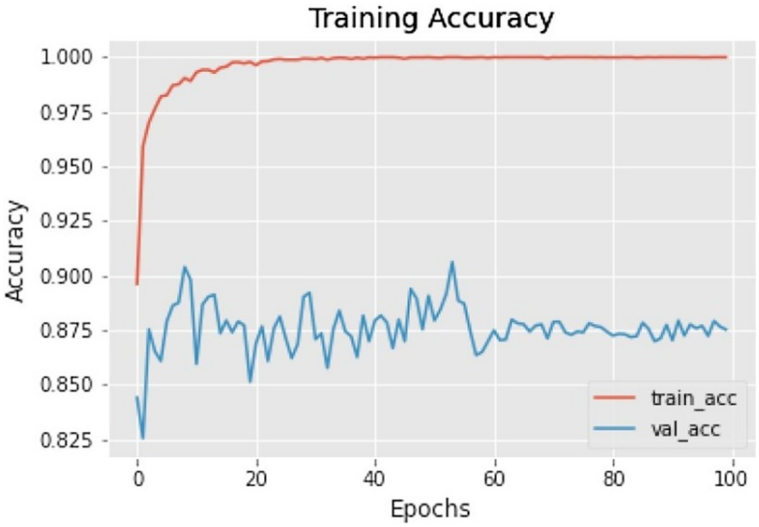


Рис. 3.21. Крива точності навчання (без збільшення даних)

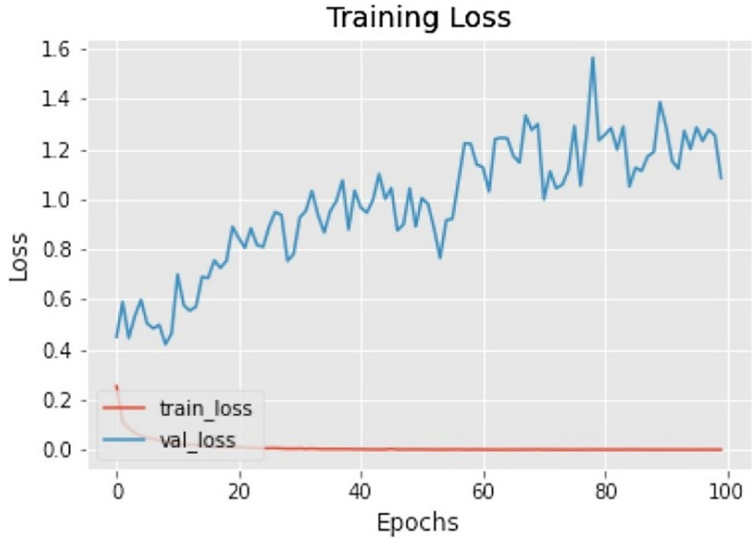


Рис. 3.22. Крива втрат на тренуванні (без збільшення даних)

Рис. 3.23. Крива точності навчання

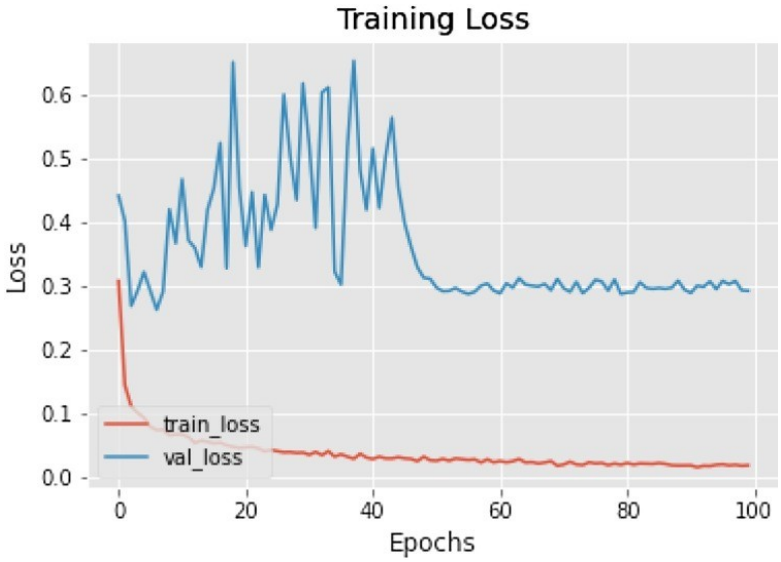


Рис. 3.24. Крива втрат на тренуванні

Appendix A Appendices

Appendix A – *Software code for working with system forms with Django*

ВИСНОВКИ

В представленій кваліфікаційній роботі було розроблено програмне забезпечення розпізнавання людини у масці.

Програмне забезпечення написано мовою *Python* з використанням відкритих для загального використання технічних рішень і фреймворків. Запропонований у роботі підхід використовує глибоке навчання, *TensorFlow*, *Keras* і *OpenCV*.

Перший розділ показав, що практично всі алгоритми розпізнавання проходять подібні один до одного етапи на шляху до розпізнавання:

1) виявлення зони обличчя;

2) визначення антропометричних точок;

3) виправлення спотворень;

4) формування вектор обличчя.

Виявлення зон обличчя та антропометричних точок передбачає використання алгоритмів сегментації Технології глибокого навчання суттєво покращили та спростили алгоритми семантичної сегментації, проклавши шлях для більш широкого застосування у реальному житті.

В другому розділ було визначено та описано методи використання нейронних мереж для зображення стилю зображення.

Просте застосування одношарової НМ (званою автоасоціативною пам'яттю) полягає в навчанні мережі та її здібностях відновити зображення, що подаються. Подаючи на вхід тестову картинку і обчислюючи якість реконструйованого зображення, можна зробити висновок, наскільки мережа розпізнала вхідне зображення. Позитивні можливості цього методу полягають в тому, що система може розпізнати спотворені і знівечині зображення, але для більш практичних цілей він не підходить. Тому було більш детально розглянута багатошарова ШНМ.

В третьому розділі було проведено проектування ПЗ та реалізовано основні класи та методи для визначення наявності медичної маски на обличчі.

Було розроблено та програмно реалізовано алгоритми повного конвеєру розпізнавання образів.

Розроблена модель може ідентифікувати обличчя навіть на розмитому фоні, і це викликає захоплення. Було помічено, що стосовно людини, яка стоїть попереду, модель не настільки впевнена (38 % у чіткій області) у порівнянні з прогнозом для людини відразу за нею (100 % у розмитій області). Це може бути пов'язане з якістю навчального набору даних, таким чином модель певною мірою схильна до впливу від якості зображення.

Середня точність моделі становить «93%» для передбачення того, чи носить людина маску чи ні в наборі даних перевірки.

Розроблену модель можна використовувати в цілях безпеки та інформування. Розроблений підхід передбачає виділення окремих зображень з потокового відео або отримання окремих зображень для визначення обличчя з використанням для класифікатора архітектури *MobilenetV*2. Розроблене програмне забезпечення не використовує жодних змінених наборів даних з замаскованими зображеннями, що було обумовлено погіршенням роботи моделі при введенні додаткових навчальних наборів з замаскованими зображеннями.

Базовий набір даних складається з 4100 зображень, які можна розділити на два класи: 2165 зображень з маскою та 1935 зображень без маски. Використані зображення були отримані з наступних відкритих джерел: *Bing Search API, Kaggle, RMFD.*

Досягнуто точність визначення обличь з маскою близько 93%. Було визначено великий вплив якості зображення на визначення обличь з маскою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Lawrence S., Giles C.L., Tsoi A.C., Back A.D. Face recognition*: *A convolutional neural-network approach*. *IEEE Transactions on Neural Networks*. 1997;8(1): 98-113.
2. *Li H., Lin Z., Shen X., Brandt J., Hua G. A convolutional neural network cascade for face detection*. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015:5325-5334.
3. *Opitz M., Waltner G., Poier G., Possegger H., Bischof H. European conference on computer vision*. *Springer*; *Cham*: 2016. *Grid loss*: *Detecting occluded faces*; *pp*. 386–402. *October*.
4. *Li Y., Sun B., Wu T., Wang Y. European conference on computer vision*. *Springer*; *Cham*: 2016. *Face detection with end-to-end integration of a convnet and a* 3*d model*; *pp*. 420-436. *October*.
5. *Li C., Wang R., Li J., Fei L. Recent trends in intelligent computing, communication and devices*. *Springer*; *Singapore*: 2020. *Face detection based on YOLOv*3; *pp*. 277-284.
6. *Ben Krose and Patrick van der Smagt. An introduction to Neural Networks*. (1996). – 126.
7. *Nguyen H. Fast object detection framework based on mobilenetv*2 *architecture and enhanced feature pyramid*. *Journal of Theoretical and Applied Information Technology*. 2020; 98 (05).
8. *Wang Z., Wang G., Huang B., Xiong Z., Hong Q., Wu H. Chen H*. 2020. *Masked face recognition dataset and application*. *arXiv preprint arXiv*: 2003.09093.
9. *Ojala T., Pietikainen M., Maenpaa T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2002;24(7):971–987.
10. Крижевский А., Суцкевер И., Джеффри Э. Хинтон Классификация *Imagenet* с глубокими сверточными нейронными сетями», «Достижения в системах обработки нейронной информации», 2012, с. 1097–1105.

11. Довідкова сторінка програми *CuneiForm*. Режим доступу: *https*://*launchpad.net*/*cuneiform*, вільний.
12. Інформацією про розпізнавач тексту *FineReader*. Режим доступу: *https*://*www.abbyy.com*/*ru-ru*/*download*/*finereader*/, вільний.
13. Ліндсей П., Нордман Д. Переробка інформації у людини. – М.: Мір, 1974. – 550 с.
14. Літюк В.И. Методичний посібник № 2231 частина 3 «Методи розрахунку і проектування цифрових багатопроцесорних пристроїв обробки радіосигналів», Таганрог, 1995, 48 с.
15. Мисюрёв А.В. Использование искусственных нейронных сетей для распознавания рукопечатных символов. В сб. "Интеллектуальные технологии ввода и обработки информации", М.: Эдиториал УРСС, 1998., 142 *с*.
16. Молинаро Энтони. *SQL*. Сборник рецептов / Энтони Молинаро. – *O'Reilly*, 2009. – 672 с.
17. Мэтью Д. Цейлер и Роб Фергус, Визуализация и понимание сверточных сетей» в *Computer Vision*. 2014, стр. 818–833, *Springer*.
18. Мэтью Д. Цейлер, Грэм У. Тейлор и Роб Фергус, «Адаптивные деконволюционные сети для изучения функций среднего и высокого уровня», в Международной конференции *IEEE* по компьютерному зрению (*ICCV*), 2011, стр. 2018–2025.
19. Нейронні мережі, їх застосування та основні функції. Режим доступу: *https*://*clck.ru*/*GLKYq*, вільний.
20. Оптичне розпізнавання символів (*OCR*), аналіз. Режим доступу *https*://*www.nicomsoft.com*/*optical-character-recognition-ocr-how-it-works*/, вільний.
21. Путятин Е.П., Аверин С.И. Обработка изображений в робототехнике. – М: Машиностроение, 2010. – 320 с
22. Структура програмного стеку бібліотеки *Tensorflow*. Режим доступу: *https*://*www.tensorflow.org*/*images*/*tensorflow_programming_environment.png*, вільний.

23. Шар відкидання в нейронних мережах. Режим доступу: *https*://*leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/ dropout_layer.html*, вільний.
24. Штучний інтелект. Системи спілкування та експертні системи. Кн. 1 / Під ред. Э.В.Попова. – М.: Радіо та зв'язок, 1990. – 461 с.
25. Щепин Е.В., Непомнящий Г.М. К топологическому подходу в анализе изображений. Геометрия, топология и приложения (Межвузовский сборник научных трудов). Москва, Мин. высшего и средн. спец. образ. РСФСР, Московский институт приборостроения, 1990., 315 *с*.
26. Якимович С. Р. Розпізнавання обличь у масці. Журнал Сучасна Спеціальна Техніка. – Київ 2020. Режим доступу: http://suchasnaspetstehnika.com.
27. ДСТУ 3008-95 Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.
28. НД ТЗІ 1.1-003-99. Термінологія у області захисту інформації в комп’ютерних системах від несанкціонованого доступу. // Департамент спеціальних телекомунікаційних систем і захисту інформації Служби безпеки України. – Київ, 1999.

Додаток А

Програмний код для роботи форм системи з *Django*

```
SECRET_KEY = 1
INSTALLED_APPS = (
    # default
    'django._miscontrib.auth',
    'django._miscontrib.contenttypes',
    'django._miscontrib.sessions',
    # extra
    'data_importer',
    'example',
    'tests',
)

DATABASES = {
    'default': {
        'ENGINE': 'django._misdb.backends.sqlite3',
        'NAME': 'test.sqlite'
    }
}

TEMPLATES = [
    {
        'BACKEND':
'django._mistemplate.backends.django._misDjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
    },
]
```

```
ROOT_URLCONF = 'example.urls'

EMAIL_BACKEND = 'django._miscore.mail.backends.console.EmailBackend'
from django._misdb.models.signals import post_delete
from data_importer.models import FileHistory
import os
from django._misconf import settings


def delete_filefield(sender, instance, **kwargs):
    """
    Automatically deleted files when records removed.
    """
    has_delete_config = hasattr(settings, 'DATA_IMPORTER_HISTORY')

    if has_delete_config and settings.DATA_IMPORTER_HISTORY == False:
        if os.path.exists(instance.filename.path):
            os.unlink(instance.filename.path)


post_delete.connect(delete_filefield, sender=FileHistory)
```