

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
РОЗПІЗНАВАННЯ ЕМОЦІЙ ЛЮДЕЙ НА ЗОБРАЖЕННІ (ВІДЕО)
У МОБІЛЬНОМУ ЗАСТОСУНКУ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS
FOR RECOGNIZING PEOPLE'S EMOTIONS IN IMAGES (VIDEOS)
IN A MOBILE APPLICATION

Виконав(ла): студент(ка) 2 курсу, групи КНТ-213м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

СУР О.О.

(ПРІЗВИЩЕ та ініціали)

Керівник СУББОТІН С. О.

(ПРІЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М. О.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

СУРА Олексія Олександровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів розпізнавання емоцій людей на зображенні (відео) у мобільному застосунку.

Research and Software Implementation of Methods for Recognizing People's Emotions in Images (Videos) in a Mobile Application

керівник проєкту (роботи) д.т.н., проф., СУББОТІН Сергій Олександрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 07 ” жовтня 2024 року № 398

2. Строк подання студентом проєкту (роботи) 01 грудня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Побудова та оцінка якості моделей. 3. Розробка програмних засобів розпізнавання емоцій. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СУББОТІН С.О., зав. кафедрою		
Нормоконтролер	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “ 09 ” вересня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2-3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4-5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7-8 тижні	Розділ 3
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	10-11 тижні	Додатки
8	Нормоконтроль та рецензування.	12 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис)

Олексій СУР

(Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Сергій СУББОТІН

(Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
89 с., 4 табл., 51 рис., 3 дод., 30 джерел.

ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, МОБІЛЬНІ ЗАСТОСУНКИ,
РОЗПІЗНАВАННЯ ЕМОЦІЙ, TENSORFLOW, KERAS.

Об'єктом дослідження даної роботи є процес розпізнавання об'єктів на зображеннях (відеопотоці).

Предмет дослідження: методи розпізнавання емоцій людини у відеопотоці.

Мета роботи – аналіз методів розпізнавання емоцій людини на зображеннях (відеопотоці) та зменшення витрат обчислювальних ресурсів при використанні моделей у мобільному додатку.

Результати. В роботі розглянуто задачу розпізнавання емоцій. Здійснено аналіз методів побудови моделей розпізнавання емоцій обличчя людини на основі нейронних мереж на основі ключових точок обличчя та згорткових нейронних мереж. Також було визначені основні характеристики побудованих моделей та здійснено їх порівняння.

Висновки. У даному проєкті (роботі) створено та тренувано модель на основі штучного інтелекту з використанням CNN, що може бути розміщена на мобільних пристроях.

Галузь використання – побудована модель може бути використана у багатьох різноманітних аспектах життя людини.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 89 pages, 4 tables, 51 figures, 3 appendixes, 30 sources.

CONVOLUTIONAL NEURAL NETWORKS, MOBILE APPLICATIONS, EMOTION RECOGNITION, TENSORFLOW, KERAS.

The object of this study is the process of recognizing objects in images (video streams).

The subject of the research are the methods for recognizing facial emotions in video streams.

The aim of this paper is to analyze methods for recognizing facial emotions in images (video streams) and to reduce computational resource when using models in a mobile application.

Results. The work examines the problem of emotion recognition. An analysis of methods for building facial emotion recognition models based on neural networks, using facial landmarks and convolutional neural networks, was carried out. The main characteristics of the developed models were identified, and their comparison was conducted.

Conclusions. In this project, a model based on artificial intelligence using CNN was created and trained, which can be deployed on mobile devices.

Practical application of the results the field of use is facial recognition in many different aspects of human life.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз предметної області	11
1.1 Розпізнавання обличчя і емоцій людини.....	11
1.2 Основні проблеми пов’язані з розпізнаванням емоцій	12
1.2.1 Проблема набору даних.....	13
1.2.2 Похибки в розпізнанні неповністю виражених емоціях.....	13
1.3 Методи розпізнавання емоцій людини.....	15
1.3.1 Метод ключових точок.....	15
1.3.2 Згорткові нейронні мережі	17
1.4 Аналіз існуючих програм.....	22
1.4.1 FaceReader	22
1.4.2 EmoDetect	23
1.4.3 the FacE Analysis System.....	24
1.5 Постановка задачі розпізнавання емоцій	26
1.6 Висновки до розділу 1	27
2 Побудова та оцінка якості моделей.....	29
2.1 Оцінка якості моделей на базі класифікації ключових точок	29
2.2 Побудова та оцінка якості моделей на основі CNN	30
2.2.1 Згорткова мережа ResNet-50	31
2.2.2 Згорткова мережа VGG	39
2.2.3 Оптимізована архітектура мережі VGG	40
2.2.4 Порівняння якості побудованих моделей.....	44
2.3 Висновки до розділу 2	45
3 Розробка програмних засобів розпізнавання емоцій.....	46
3.1 Вимоги для розробки мобільного додатку.....	46
3.2 Вибір інструментів розробки програми	48
3.3 Основні рішення щодо реалізації класів	50

3.4 Інтеграція з Apple CoreML	53
3.5 Адаптування моделі до використання у додатку.....	54
3.6 Висновки до розділу 3	57
4 Експлуатація, тестування та експериментальне дослідження програми	58
4.1 Експлуатація програми	58
4.2 Методика та результати тестування	59
4.3 Огляд та порівняння датасетів для розпізнавання емоцій	62
4.3.1 Датасет AffectNet	62
4.3.2 Датасет EMOTIC.....	63
4.3.3 Датасет FER2013.....	64
4.3.4 Порівняння датасетів.....	65
Висновки.....	67
Перелік джерел посилань.....	69
Додаток А Шари CNN на базі архітектури VGG	72
Додаток Б Фрагменти коду програм	74
Додаток В Слайди презентації	83

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AAM	– Active Appearance Models;
AI	– Artificial Intelligence;
API	– Application Programming Interface;
CNN	– Convolutional Neural Network;
IDE	– Integrated Development Environment;
EGM	– Elastic Graph Matching;
GUI	– Graphic User Interface;
ML	– Machine Learning;
MLP	– Multi Layer Perceptron;
MVP	– Minimum viable product;
LFW	– Labeled Faces in the Wild;
PCA	– Principal Component Analysis;
КЗ	– комп'ютерний зір;
РО	– розпізнавання обличь;
ПЗ	– програмне забезпечення.

ВСТУП

У зв'язку з великою кількістю камер у всіх аспектах життя людини, все більш актуальним стають розробки в розпізнаванні осіб. Задача розпізнавання обличч є ключовою як для розвитку інтелектуальних середовищ, так і для впровадження систем безпеки. Проте, якщо людина здатна за частки секунди ідентифікувати та порівняти обличчя з тими, що зберігаються в її пам'яті, то машина, без відповідної технології розпізнавання, не зможе відрізнити обличчя людини від іншого об'єкта.

Для початку комп'ютер повинен не просто розпізнати людину, але і зрозуміти, що знаходиться перед ним, особа чи щось інше. А надалі так само, як і у людини, його потрібно навчити впізнавати обличчя, орієнтуючись на індивідуальні риси: відстань між очей, їх колір, висота губ і їх ширина, розмір носа. При цьому, ракурс, під яким камера фіксує обличчя людини, освітлення та наявність сторонніх предметів на обличчі мають значний вплив.

В даний час спостерігається неминущий інтерес до проблеми розпізнавання обличчя та емоцій. Практичне значення отриманих результатів полягає в їхньому потенціалі для значного покращення якості життя людей у різних сферах. Зокрема, системи розпізнавання емоцій у відео можуть бути корисними в медицині, допомагаючи покращувати процес діагностики та лікування пацієнтів шляхом своєчасного виявлення емоційних розладів, наприклад, депресії чи тривоги. Це дозволить своєчасно надавати необхідну допомогу, що може мати критичне значення для ефективного лікування.

Крім того, алгоритми розпізнавання емоцій можуть бути впроваджені в системи безпеки, де вони допоможуть аналізувати поведінку людей у громадських місцях. Це дозволить виявляти потенційно небезпечні ситуації та попереджати злочини, тим самим підвищуючи рівень безпеки суспільства.

В автомобільній індустрії використання таких систем може сприяти зменшенню кількості дорожньо-транспортних пригод через виявлення ознак

втоми або роздратування водія і своєчасного попередження про необхідність зробити перерву, що сприятиме безпеці на дорогах.

У сфері освіти системи розпізнавання емоцій можуть бути корисними для викладачів, допомагаючи їм краще розуміти емоційний стан студентів. Це дозволить виявляти моменти, коли студенти відчують труднощі або втрачають інтерес до навчання, і своєчасно надавати їм необхідну підтримку.

У маркетингу та обслуговуванні клієнтів впровадження таких систем допоможе компаніям краще розуміти реакції клієнтів на продукти чи послуги, що дозволить удосконалювати маркетингові стратегії та підвищувати якість обслуговування. Соціальні мережі та комунікації також можуть виграти від використання систем розпізнавання емоцій, оскільки це сприятиме створенню більш персоналізованого контенту та покращенню взаємодії між користувачами.

Отже, створення систем розпізнавання емоцій людини на відео відкриває широкі можливості для їх використання, сприяючи підвищенню якості життя, розвитку сучасних технологій, а також забезпеченню безпеки та зручності в повсякденному житті.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Розпізнавання обличчя і емоцій людини

Всі сучасні технології розпізнавання осіб використовують системи, які навчаються за допомогою тестових зображень. Для навчання використовуються бази з зображеннями, що містять особи, і не містять особи. Кожен фрагмент досліджуваного зображення характеризується як вектор ознак, за допомогою якого класифікатори (алгоритми для визначення об'єкта в кадрі) визначають, є дана частина зображенням особи чи ні [1]-[2].

Розпізнавання емоцій за виразом обличчя – об'єктивний метод оцінки реакції людини на деяке явище або іншу людину. Емоції відіграють величезну роль в житті людини і міжособистісному спілкуванні. Вони можуть бути виражені різними способами: мімікою, позою, руховими реакціями, голосом і вегетативними реакціями (частота серцевих скорочень, артеріальний тиск, частота дихання). Однак найбільшою виразністю володіє обличчя людини: подивившись на зморщений ніс, насуплені брови або широку посмішку, можна багато чого сказати про людину й її емоційний стан [3].

Але правильно інтерпретувати емоцію людини не завжди просто. Сучасні алгоритми машинного навчання демонструють точність в 55% для розпізнавання виразу обличчя на зображеннях [4].

Основний алгоритм автоматичного розпізнавання емоцій складається з наступних етапів [4]:

Отримання зображення з обличчям людини.

- обробка отриманого зображення та виділення людини на зображенні;
- розпізнавання елементів обличчя людини;
- виділення ключових точок на обличчі;
- класифікація емоцій.

Однак емоції людини мають не статичний, а динамічний характер, вони досить швидко змінюються, і сама швидкість їх зміни теж може служити показником емоційного стану.

Модель розпізнавання емоцій на відео схожа на розпізнавання емоцій на зображеннях, але має свої особливості. По-перше, вона починається з отримання необхідної інформації з відео: спочатку отримуються кадри, а потім розпізнаються обличчя і виконується корегування для кожного окремого кадру. Крім того, пропонується об'єднувати кадри з плином часу, оскільки тимчасова коваріація захоплює необхідну динаміку руху обличчя [5].

Отримавши кадр з відео-потoku, може використовуватись той самий алгоритм аналізу емоцій, що використовувався при розпізнаванні емоцій з зображення, бо окремий кадр вважається окремим зображенням [5].

1.2 Основні проблеми пов'язані з розпізнаванням емоцій

Проблеми, пов'язані з розпізнаванням і ідентифікацією осіб по відеофіксації в режимі реального часу, можна класифікувати на наступні:

- проблеми, пов'язані з навчанням. Навчання систем розпізнавання та ідентифікації осіб по відеофіксації в режимі реального часу реалізується за допомогою поділу і класифікації окремих об'єктів із зазначенням їх приналежності того чи іншого способу. У результаті навчання система розпізнавання набуває здатності адекватно реагувати на об'єкти одного або різних типів;

- проблеми, пов'язані з розпізнаванням. Після етапу навчання відбувається процес розпізнавання об'єктів, який демонструє здатність уже навченої системи до ідентифікації. Автоматизація цих процедур і становить проблему навчання розпізнавання образів. У тому випадку, коли людина сама розгадує або придумує, а потім нав'язує системі правило класифікації, проблема розпізнавання вирішується частково, так як основну і головну

частину проблеми (навчання) людина бере на себе.

Для того, щоб система ефективно розпізнавала і ідентифікувала людей по відеофіксації в режимі реального часу необхідно забезпечити максимальну інваріантність до різних зображень осіб в умовах впливу зовнішніх факторів. Зовнішніми факторами можуть виступати: освітлення, положення голови у просторі, фон, масштаб зображення обличчя, часткове закриття обличчя (окуляри, волосся, вуса, борода) [6].

1.2.1 Проблема набору даних

Якісний набір даних є одною з основних проблем, з якою зіштовхуються фахівці з обробки даних. Набір даних повинен відповідати наступним критеріям якості для будь-якої задачі машинного або глибокого навчання:

- зображення найвищої якості: прикладні зображення, які використовують для навчання моделей повинні бути максимальної якості, чіткими та з високим розширенням для кращого аналізу та розпізнання;
- великий вміст набору даних: найкращі результати показують моделі, які були натреновані на великому обсязі даних;
- варіативність набору даних: для ефективного тренування моделі важливо використовувати набір даних з різноманітними характеристиками. Це дозволяє моделі навчитися розпізнавати та адаптуватися до різних ситуацій, забезпечуючи готовність до обробки нових даних у майбутньому. Різнманітність може включати змінні параметри, такі як різні контексти фотографій, кути зору, різноманітність національностей людей на зображеннях, їх вік та наявність інших об'єктів.

1.2.2 Похибки в розпізнанні неповністю виражених емоціях

Більшість наборів даних, розроблених для вирішення завдання щодо здійснення розпізнавання штучних емоцій на фотографіях людей. В таких

випадках людина намагається максимально передати необхідну емоцію у повному обсязі (рис. 1.1). Щодо реального життя, то люди рідко проявляють максимальний спектр власних емоцій (рис. 1.2).



Рисунок 1.1 – Програмна модель вираження емоцій [7]



Рисунок 1.2 – Приклад прояву емоцій в реальному житті [8]

Як можна побачити, ця різниця є досить значною, що ускладнює наше завдання.

1.3 Методи розпізнавання емоцій людини

1.3.1 Метод ключових точок

Алгоритм розділений на два етапи: отримання координат ключових точок і класифікація емоцій на їх основі [9].

Вхідними даними для методу є набір зображень обличчя, в якому кожному зображенню відповідає файл розмітки, що містить координати ключових точок, обраних людиною. За цими даними будується дві статистичні моделі: модель форми та модель текстури [9].

Модель форми – параметрична лінійна модель, що описує можливі варіанти положення ключових точок. Формою при цьому називається вектор координат ключових точок, яка наведена в формуле (1.1) [9]:

$$S = (x_1, y_1, x_2, y_2, \dots, x_n, y_n)^T. \quad (1.1)$$

Модель текстури – подібна модель, але описує вже можливі варіації інтенсивності пікселів. Відповідно текстурою називається вектор всіх пікселів всередині зовнішнього контуру форми, яка представлена в формуле (1.2) [9]:

$$S = (t_1, t_2, \dots, t_n)^T. \quad (1.2)$$

Оскільки на різних зображеннях кількість пікселів всередині зовнішнього контуру форми може бути різним, перед створенням моделі текстури всі зображення обличчя приводяться до єдиної усередненої форми за допомогою кусочно-афінного перетворення: безліч точок форми триангулюється, а потім кожен симплекс звичайним афінним перетворенням транслюється в нові координати [9].

Визначити емоції можна на основі аналізу декількох ключових точок. Наприклад, за допомогою комбінацій точок брів і рота (рис. 1.3).

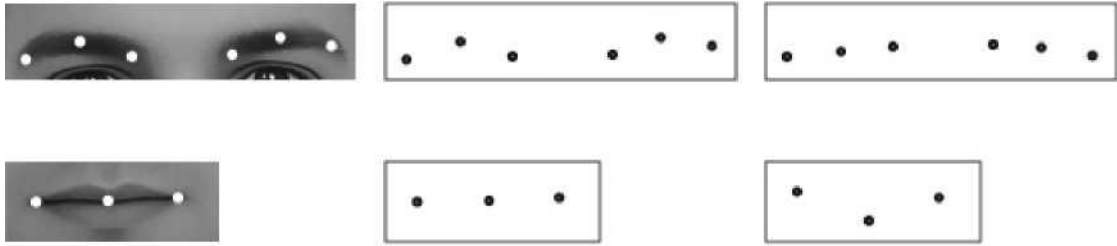


Рисунок 1.3 – Комбінація ключових точок брів і рота, відповідна їх зображенням [9]

Виділення ключових точок здійснюється наступним чином, що проілюстровано на рисунку 1.4:

- перетворення кольорового зображення до напівтонового виду;
- перетворення з напівтонової форми до бінарного вигляду;
- застосування до бінарного зображення градієнтної маски;
- локалізація ключових точок.

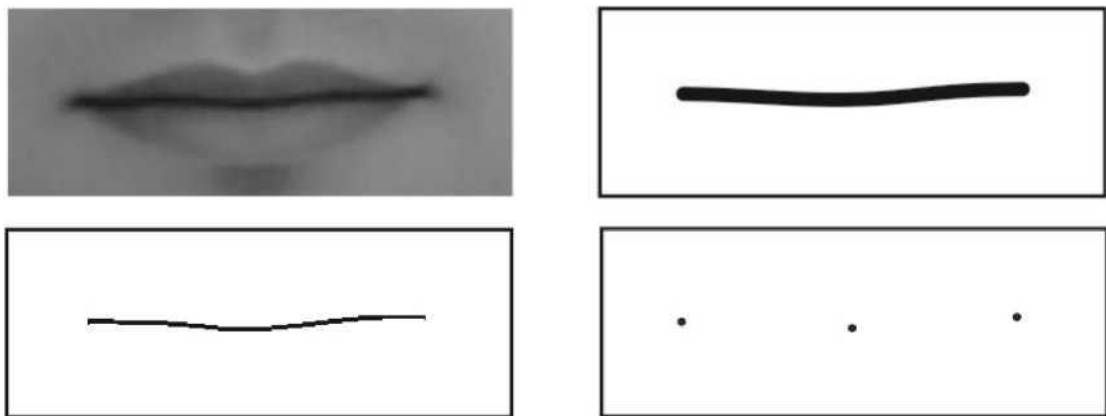


Рисунок 1.4 – Процес виділення ключових точок [9].

Перехід від кольорового зображення до напівтонового виконується на етапі виділення обличчя. Для отримання бінарного зображення використовується адаптивний поріг. Застосування градієнтної маски до бінарного зображення дозволяє отримати контурне уявлення аналізованого елемента. Локалізація ключових точок полягає в визначенні заданої кількості точок, що лежать на контурі елемента.

Після виділення ключових точок можна виконати класифікацію емоції (табл. 1.1).

Таблиця 1.1 – Класифікація емоцій за допомогою ключових точок

Емоція	Брови	Рот
Подив	Піднімаються	Відкривається
Страх	Піднімаються і зморщуються	Відкривається і розтягується
Відраза	Знижуються	Піднімається і кінці звужуються
Гнів	Знижуються і зморщуються	Відкривається або кінці знижуються
Щастя	Піднімаються	Кінці піднімаються
Смуток	Кінці знижуються	Кінці знижуються

Переваги та недоліки: До переваг методу можна віднести використання недорогого обладнання; при відповідному обладнанні є можливість розпізнавання зображень отриманих на значних відстанях. Недоліки наступні: доволі високі вимоги до освітлення, зображення особи обов'язково повинно бути фронтальним, з невеликими відхиленнями. Не враховують можливі зміни міміки обличчя.

1.3.2 Згорткові нейронні мережі

Згорткові нейронні мережі (convolutional neural network, CNN) широко застосовується в багатьох підходах комп'ютерного зору, включаючи проблему розпізнавання емоцій. На початку XXI століття в кількох дослідженнях літератури розпізнавання емоцій обличчя, встановлено, що CNN є надійним підходом у випадку змін місця розташування та варіаціями

масштабу і поводить краще, ніж багатошаровий перцептрон (MLP) у випадку раніше небачених варіацій поз обличчя [10].

Згорткова нейронна мережа являє собою особливий клас багатошарового перцептрона, який володіє двовимірною структурою і добре підходить для обробки зображень з високим ступенем інваріантності до зсуву, поворотам, масштабування і іншим спотворень вхідних даних.

На сьогоднішній день цей тип нейронних мереж демонструє високу ефективність при роботі із зображеннями та досягає найкращих результатів у задачах розпізнавання та сегментації об'єктів.

CNN має певні переваги над ШНМ:

- менша кількість налаштованих ваг;
- можливість навчання на графічних процесорах;
- розпаралелювання обчислень;
- стійкість до спотворень зображення таких як зміна масштабу, повороту та зсуву картини.

На рис. 1.5 зображена структура CNN:

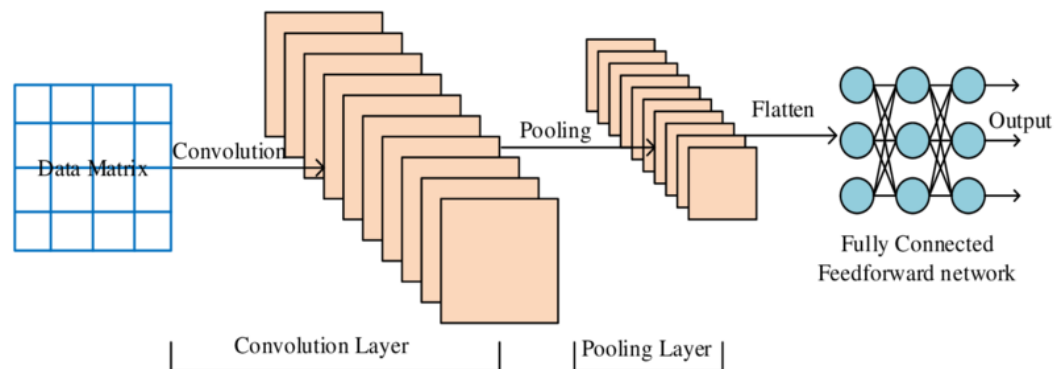


Рисунок 1.5 – Структура CNN [11]

Архітектура CNN містить три основні типи шарів, кожен з яких відіграє певну роль:

- згортковий шар (Convolution layer);

Відповідає за знаходження певних рис зображення за допомогою фільтрів, котрі виявляють певні характеристики в картинках. Локальна

кореляція забезпечує сильний зв'язок пікселів, які знаходяться поруч та саме в цьому шарі виконується операція згортки.

Згортка – операція, яка здійснюється над парою матриць $I (a \times b)$ та $K(m \times n)$ і є зваженою сумою усіх значень пікселів. Добуток цих матриць кодує ознаку графічно (рис. 1.6). Ця операція зменшує кількість інформації, яка зберігається в пам'яті, та, як наслідок, краще та швидке розпізнає зображення [10].

Формула (1.3) для визначення згортки:

$$(I * K)[a,b] = \sum_{m,n} I[a-m, b-n] * K[m,n], \quad (1.3)$$

де I – зображення;

K – фільтр;

m, n – розмірності фільтра.

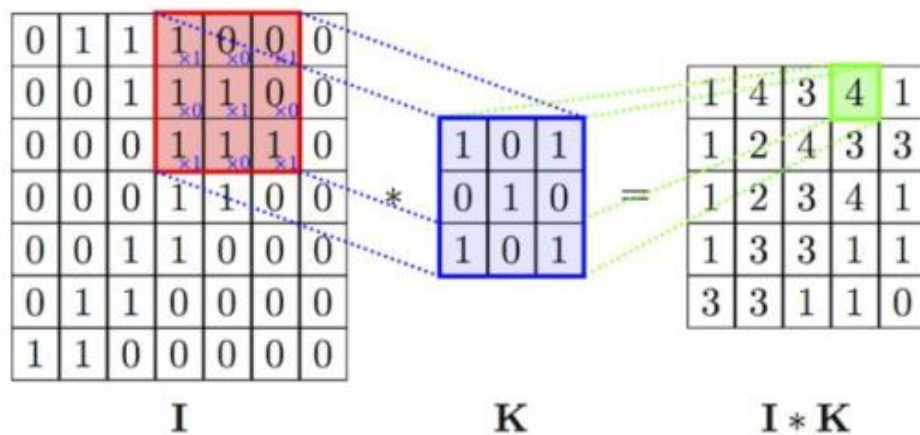


Рисунок 1.6 – Розрахунок згортки [10]

Фільтр визначає присутність потрібної характеристики в певній частині зображення. Фільтри розміщуються паралельно один одному. Чисельність фільтрів призводить до збільшення чисельності виявлених характеристик та їх надмірна кількість може призвести до перенавчання моделі. Кількість фільтрів дорівнює чисельності карт ознак, котрі є сукупністю нейронів, які обчислюють свою функцію [10].

– субкредитуєчий шар чи шар пулінгу (Pooling layer);

Виконують функцію зменшення вагів та відповідно зменшують можливість перенавчання НМ і зводять до мінімуму навантаження на обчислювальний пристрій. На вхід цей шар отримує матрицю та зменшує її розмірність за рахунок усереднення значень сусідніх пікселів. CNN утворюються за рахунок чергування шарів згортки і шарів пулінгу.

Субкредитуєчий шар зменшує розмірність картки ознак через те, що виконує усереднення для суміжних пікселів. Обрахунок карти ознак виконується за наступною формулою (1.4):

$$y_j^m(x, y) = f_j(w_{ij}^m \sum_{(g_x, g_y) \in G} y_j^{m-1}(sx + g_x, sy + g_y)), \quad (1.4)$$

де y_j^m – карта ознак;

w_{ij}^m – ваги нейронів;

s – коефіцієнт зміни розмірності.

Коефіцієнт деформації, який виконує функцію навчання мережі, може бути типу вибору аксимального елементу (max-pooling), тобто визначення максимуму для кожної області (рис. 1.7) чи типу обрання середнього значення (average pooling) і т.д. Max-pooling є найефективнішим [10].

Формула (1.5) для цього типу має вигляд:

$$y_j^m(x, y) = \max y_j^{m-1}(sx + g_x, sy + g_y), \quad (1.5)$$

де y_j^m – активаційна карта;

s – коефіцієнт зміни розмірності.

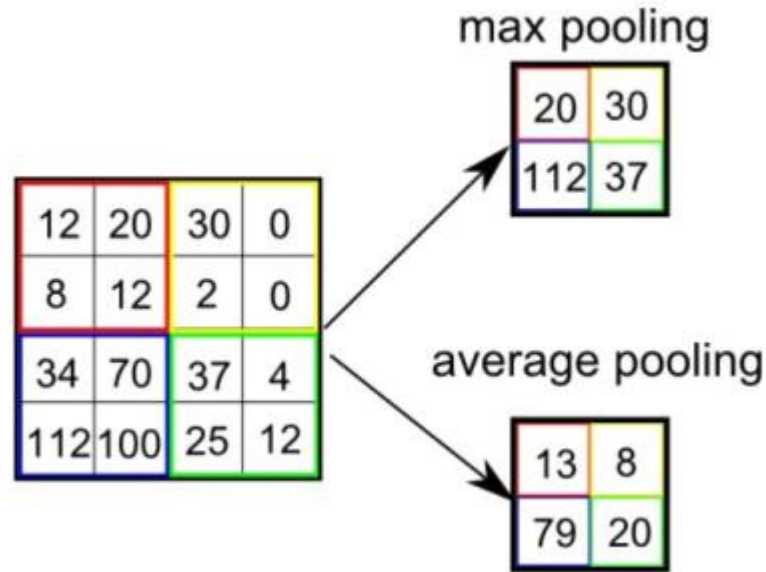


Рисунок 1.7 – Шар пулінгу [10]

– повнозв’язний шар (Fully connected layer);

Цей шар відповідає за класифікацію виявлених характеристик в вхідних зображеннях [10]. Кожний нейрон шару зв’язаний з кожним нейроном попереднього і наступного шару.

Містить локальні поля та активації нейрону. Формули (1.6) та (1.7) для їх обчислення наведені нижче:

$$v_j^m = \sum_i w_{ij}^m y_j^{m-1}, \quad (1.6)$$

$$y_j^m = f_j(v_j^m), \quad (1.7)$$

де v_j^m – локальні поля;

y_j^m – активації локальних полів.

Зазвичай, цей шар закінчується шаром softmax. Він репрезентує виходи ЗНМ по класах розпізнавання. В цьому шарі кількість блоків є рівною кількості класів.

Послідовне поєднання шарів згортки та пулінгу зменшує розмір даних і водночас збільшує кількість карт ознак.

1.4 Аналіз існуючих програм

1.4.1 FaceReader

Цей застосунок може вірно розпізнавати по виразу обличчя людини такі емоції, як «щастя», «смуток», «злість», «здивування», «страх», «відраза» і «спокій». Крім того, FaceReader здатний визначити вік, стать і расу людини. Додаток не має потреб в навчанні і додатковому налаштуванні.

У програмі реалізовані технології комп'ютерного зору. Зокрема, це метод Active Template, що полягає в накладенні деформованого шаблону на зображення обличчя. Інтерфейс застосунку FaceReader продемонстровано на рисунку 1.8 [11].

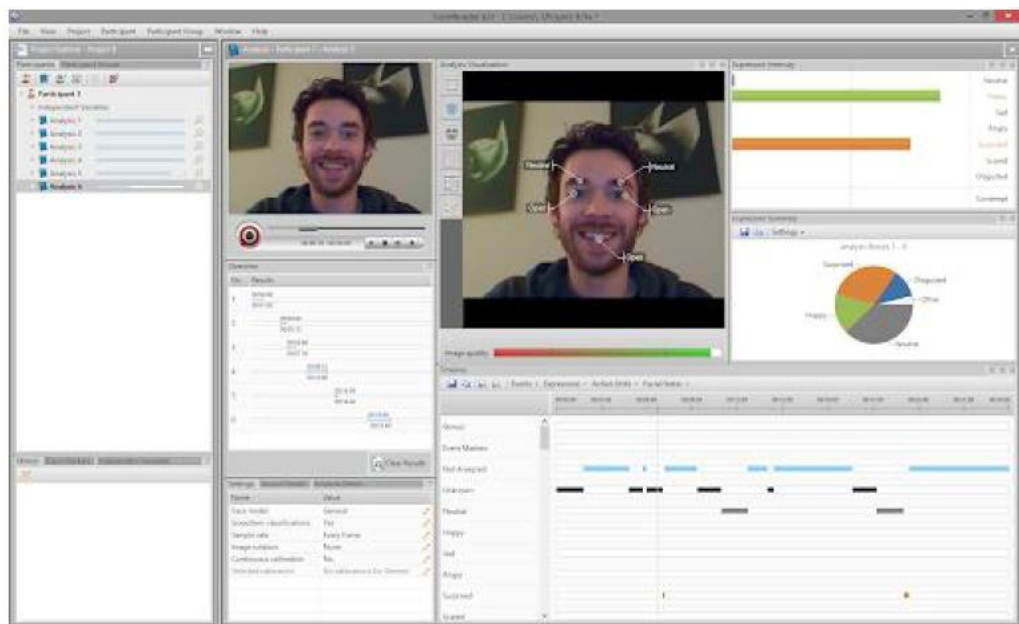


Рисунок 1.8 – Інтерфейс програми «FaceReader» [11]

Переваги даної системи:

– середній відсоток розпізнавання емоцій доволі високий і не залежить від нахилу й повороту обличчя;

- програма працює з великою кількістю форматів відеофайлів з кодеками MPEG1, MPEG2, DivX4, DivX5, DivX6, DV-AVI;
- додаток не обмежується роботою з відео, які необхідно завантажити, FaceReader працює і зі статичними зображеннями і в реальному часі, зчитуючи дані з камери користувача;
- різні можливості візуалізації: гістограми, діаграми, процентне відображення аналізованих емоцій;

Недоліки системи:

- відсоток розпізнавання емоцій і віку сильно падає при роботі з обличчями дітей віком менше 5 років;
- визначення може відбуватися неточно, якщо людина носить окуляри або має бороду;
- нейтральний стан людини не може бути визначений.

1.4.2 EmoDetect

Програмне забезпечення EmoDetect дозволяє визначити психоемоційний стан людини по виборці зображень (відео або набір файлів) [12].

Розроблений класифікатор емоцій дозволяє визначити 6 базових емоцій: радість, здивування, смуток, злість, страх, презирство. Аналіз емоцій відбувається за допомогою нейронних мереж: виконується знаходження опорних точок на обличчі людини, і класифікатор будує загальну картину на основі їх становища щодо одне одного.

Розрахунок рухових одиниць і їх класифікація відбувається за системою кодування лицьових рухів П. Екмана (FACS Action Units). Інтерфейс програми EmoDetect проілюстровано на рисунку 1.9.

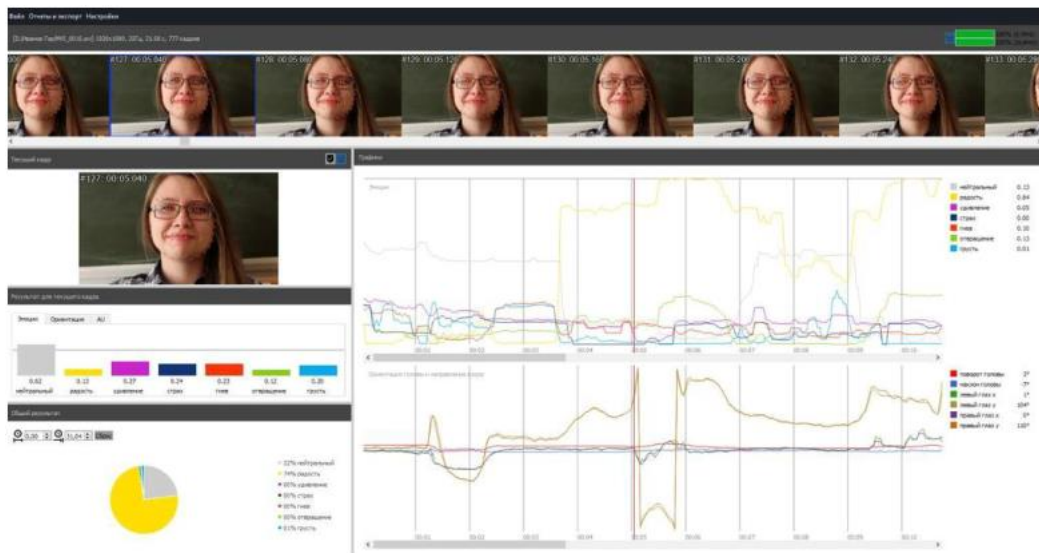


Рисунок 1.9 – Интерфейс програми «EmoDetect» [12]

Переваги системи:

- можливість визначення нейтрального виразу обличчя користувача.
- побудова графіка динамічної зміни інтенсивності емоцій випробуваного в часі і формування звіту про результати обробки відео;
- можливість запису відео-файлів з веб-камери, аналіз всіх підтримуваних операційною системою форматів;
- режим вкладок для проведення порівняльної оцінки;
- ідентифікація по обличчю в режимі реального часу.

Недоліки системи:

- обличчя, що повернуті в профіль, не можуть бути проаналізовані;
- складність аналізу емоцій людей різної раси.

1.4.3 the FacE Analysis System

Цей програмний застосунок сканує потік вхідних даних і, при виявленні в кадрі осіб, порівнює їх з заведеною базою даних на предмет наявності збігів. У разі знаходження збігів система відправляє повідомлення операторам в реальному часі [13].

Цей додаток широко використовується в системі безпеки європейських країн для виявлення підозрілих осіб в громадських місцях.

Великі компанії ж використовують його для ідентифікації важливих клієтів. Рядові фірми використовують дану систему для розпізнавання обличчя своїх співробітників і виявлення незареєстрованих відвідувачів, використовуючи бази даних з фотографіями своїх підлеглих.

Також дана розробка допомагає аналізувати стан людини й повідомляє в разі отримання підозрілого результату. Інтерфейс системи eMotion Software можна побачити на рисунку 1.10 [13].

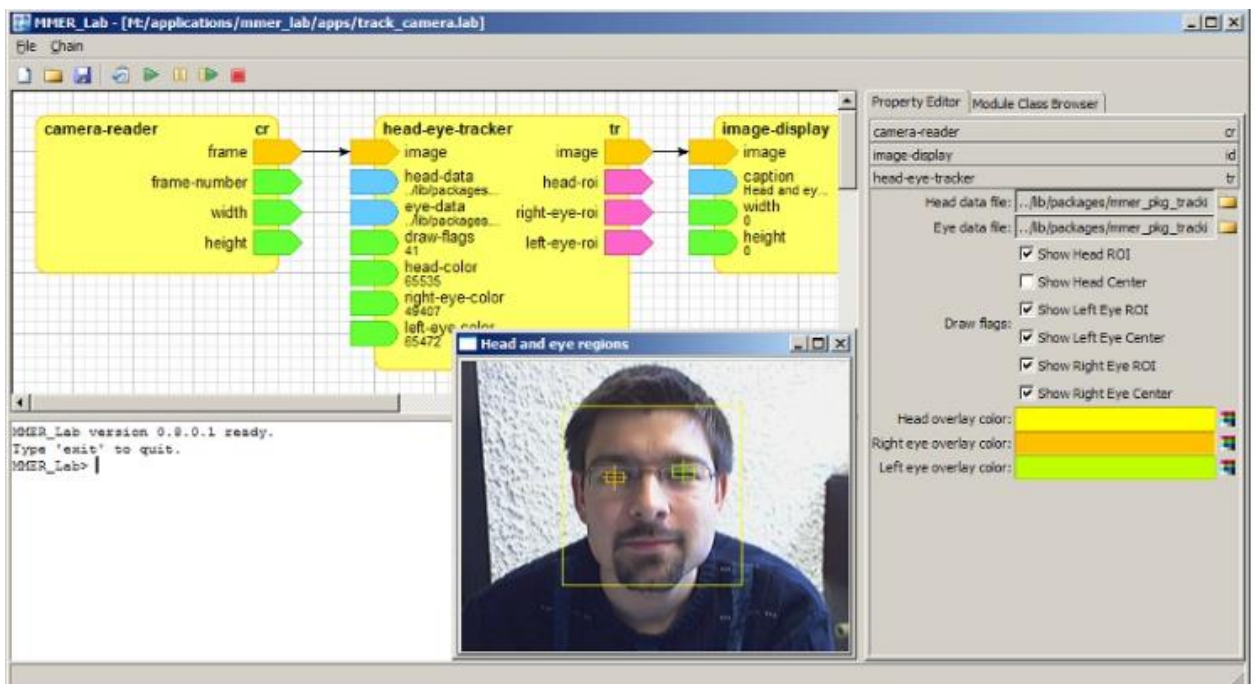


Рисунок 1.10 – Інтерфейс програми «eMotion Software» [13]

Система використовує такі етапи розпізнавання обличчя:

- детектування особи і силуету на зображенні;
- виправлення візуальних спотворень;
- витяг характеристик особи;
- верифікація або ідентифікація особи.

Переваги програми:

- робота в реальному часі, можливість стеження за особою на декількох потоках даних одночасно;
- у режимі реального часу особа порівнюється з еталонами, занесеними в базу даних;

- відео в ручному і автоматичному режимі;
- застосування C++ API і Web Services API;
- можливість масштабованості по таким критеріям: розмір, кількість відеопотоків і число видимих осіб на зображенні;
- недоліки програми:
- неможливість проведення обчислень для осіб, що мають відхилення від фронтального положення більш ніж 15 градусів;
- світлочутливість. Неможливість розпізнавання особи, при попаданні на нього тіней і відблисків.

1.5 Постановка задачі розпізнавання емоцій

Підсумовуючи все вище зазначене, сформулюємо постановку задачі: перед нами стоїть задача класифікації емоцій людини за 7 класами на основі відеопотоку.

Таким чином ми приходимо до предмета дослідження, а саме якості методів розпізнавання емоцій у відеопотоці. У даній роботі було вибрано нейронні мережі, а саме нейронну мережу на основі технології згорткової нейронної мережі.

Перед нами будуть стояти наступні задачі, а саме:

- підготувати вибірку даних;
- перевести отримані дані у чорно-білий канал та масштабувати їх до одного розміру;
- побудувати модель на основі нейронних мереж, використовуючи різні підходи;
- оцінити якість отриманих моделей, а саме їх точність, повноту та розміри;
- вибрати найкращу модель.
- конвертувати її у формат, і за розміром який може буди застосований у мобільній розробці

- створити мобільний додаток, який буде по відеопотоку отриманому з камери розпізнавати емоції людини;
- дослідити отримані результати та зробити висновки.

1.6 Висновки до розділу 1

В даному розділі було розглянуто актуальність задачі розпізнавання емоцій, визначено предмет та об'єкт дослідження, сформульовано основні задачі роботи.

На сьогоднішній день існує значна кількість продуктів, які намагалися вирішити проблему розпізнавання емоцій у людини. Незважаючи на широкий асортимент додаткових функцій та привабливе оформлення, більшість з них має свої недоліки, що не дозволяє вважати їх еталоном та стверджувати, що проблема вже вирішена.

Основні недоліки таких продуктів включають наступне:

- низький рівень точності розпізнавання;
- неспроможність розпізнати емоції у дітей;
- значне погіршення точності при нахилі обличчя під кутом більше ніж 15 градусів;
- деякі системи споживають багато ресурсів і часу для аналізу емоцій, що ускладнює розпізнавання емоцій у режимі реального часу.

Ці обмеження стимулюють розробку нового, більш якісного продукту, який зможе вирішити ці проблеми більш ефективно.

А саме було вибрано 7 емоцій людини як об'єкт дослідження. В якості предмету було обрано якість методів розпізнавання емоцій людини.

Проблема ідентифікації емоцій людини може бути вирішена, з використанням сучасних методів розпізнавання на основі згорткових мереж.

Таким чином було визначено наступні задачі роботи, такі як: підготовка первинних даних у вигляді зображень та відеопотоці, виділення

обличчя з даних, побудова нейронних мереж для розпізнавання емоцій, а також порівняння їх якості.

Було обговорено проблеми, які існують перед дослідником при побудові бази даних для розпізнавання емоцій.

2 ПОБУДОВА ТА ОЦІНКА ЯКОСТІ МОДЕЛЕЙ

У цьому розділі досліджується оцінка та порівняння різних моделей для класифікації та розпізнавання образів, зокрема на основі класифікації ключових точок, а також глибоких згорткових нейронних мереж.

Оцінка якості моделей є важливою для розуміння їх ефективності та потенціалу в реальних застосунках, таких як розпізнавання обличчя, емоцій або інших об'єктів на зображеннях [14].

Моделі на основі CNN, зокрема згорткові мережі ResNet-50 та VGG, є одними з найбільш популярних для таких завдань завдяки своїй гнучкості та здатності обробляти складні дані.

2.1 Оцінка якості моделей на базі класифікації ключових точок

Найбільш простий спосіб визначення емоції зображення обличчя заснований на класифікації ключових точок (facial landmarks), координати яких можна отримати, використовуючи різні алгоритми PDM, CML, AAM, DPM або CNN (рис. 2.1). Зазвичай розмічають від 5 до 68 пікселів, прив'язуючи їх до положення брів, очей, губ, носа, щелепи, що дозволяє частково захопити міміку. Нормалізовані координати точок можна безпосередньо подати до класифікатора (наприклад, SVM або Random Forest) та отримати базове рішення. При цьому становище осіб у своїй вибірці має бути вирівняно фронтально [15].

Просте використання координат без візуальної компоненти (рис. 2.2) призводить до істотної втрати корисної інформації, тому для поліпшення системи в цих точках обчислюють різні дескриптори: LBP, HOG, SIFT, LATCH та ін. емоцій [15].

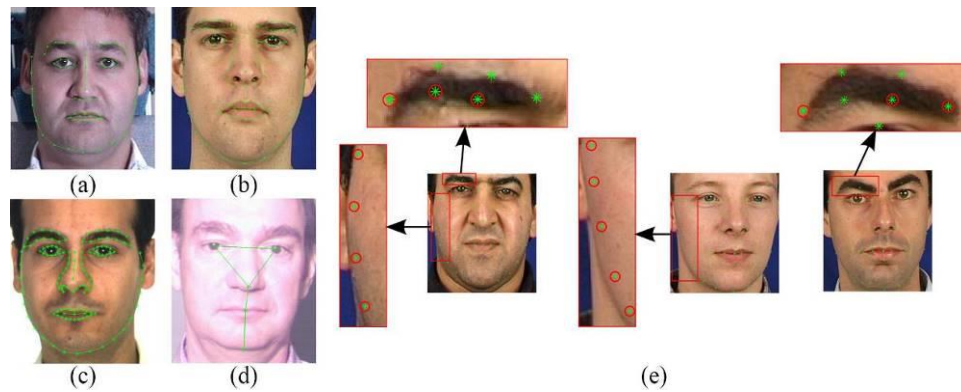


Рисунок 2.1 – Приклад визначення ключових точок [16]

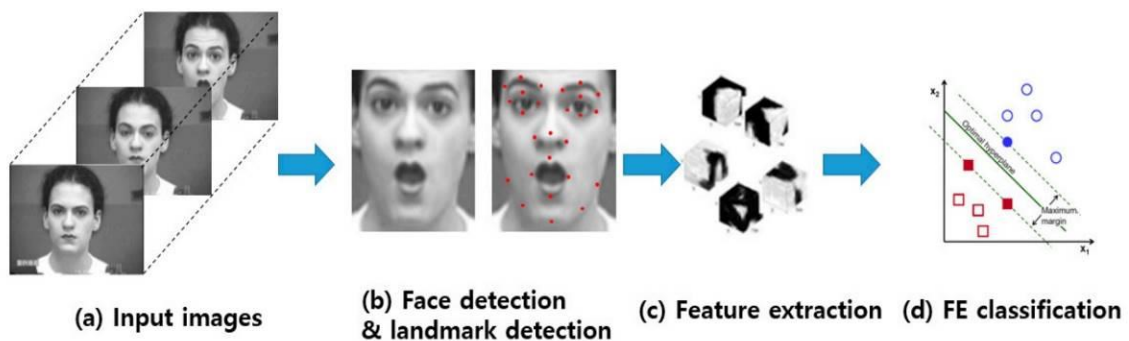


Рисунок 2.2 – Процес класифікації емоцій за координатами на обличчі [17]

Однак такий підхід вже вважається застарілим, оскільки відомо, що глибокі згорткові мережі є найкращим вибором для аналізу візуальних даних.

2.2 Побудова та оцінка якості моделей на основі CNN

Продуктивність і ефективність CNN визначається її архітектурою. Це включає структуру шарів, те, як елементи розроблені та які елементи присутні в кожному шарі. Було створено багато CNN, але нижче наведено деякі з найвідоміших, які за своїми характеристиками підходять для вирішення поставлених задач.

2.2.1 Згорткова мережа ResNet-50

ResNet-50 – це архітектура CNN, яка належить до сімейства ResNet (Residual Networks), серії моделей, призначених для вирішення проблем, пов'язаних із навчанням глибоких нейронних мереж. ResNet-50, розроблений дослідниками Microsoft Research Asia, відомий своєю глибиною та ефективністю в задачах класифікації зображень. Архітектури ResNet мають різну глибину, наприклад ResNet-18, ResNet-32 тощо, причому ResNet-50 є варіантом середнього розміру.

Основною проблемою, яку вирішив ResNet, була проблема деградації глибоких нейронних мереж. Коли мережі стають глибшими, їх точність насичується, а потім швидко погіршується. Ця деградація спричинена не надмірною підготовкою, а радше труднощами оптимізації тренувального процесу [18].

На рисунку 2.3 наведені графіки порівнянь: помилка навчання (ліворуч) і помилка тесту (праворуч) на датасеті CIFAR-10 з 20-шаровими та 56-шаровими упрости за архітектурою мережами. [18].

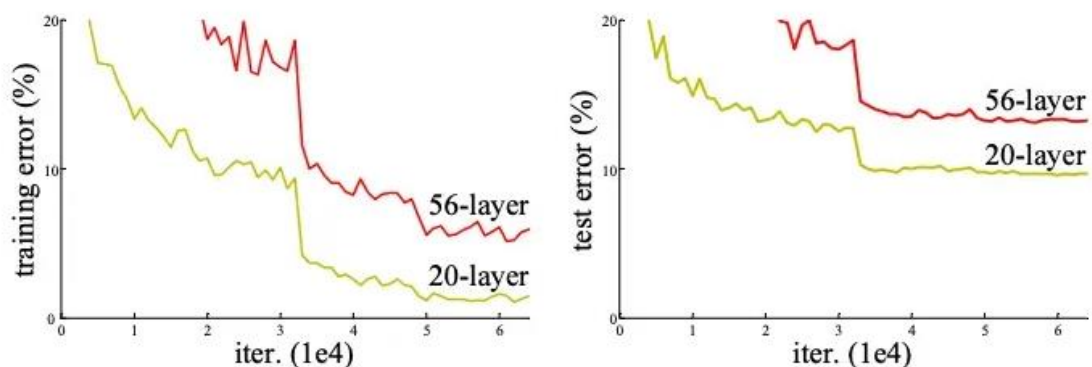


Рисунок 2.3 – Порівняння результатів навчання і тестування мереж з різною кількістю шарів [18]

ResNet вирішила цю проблему за допомогою залишкових блоків, які забезпечують прямий потік інформації через з'єднання пропуску, пом'якшуючи проблему зникнення градієнта. Залишковий блок, який

використовується в ResNet-50, називається блоком залишкового вузького місця. Цей блок має наступну архітектуру (рис. 2.4).

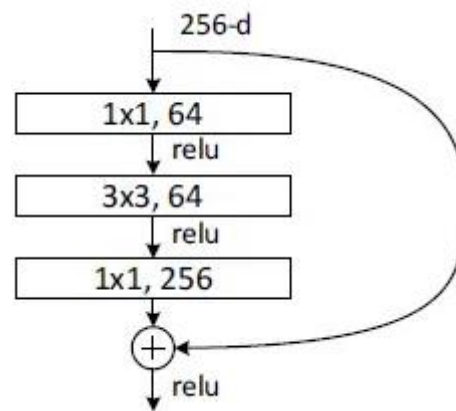


Рисунок 2.4 – Залишковий блок вузького місця для ResNet-50 [18]

Ось розбивка компонентів у залишковому блоці. Функція активації ReLU (Rectified Linear Unit) застосовується після кожного згорткового шару та шарів пакетної нормалізації. ReLU пропускає лише додатні значення, вносячи в мережу нелінійність, яка є важливою для того, щоб мережа вивчала складні закономірності в даних.

Шари згортки вузьких місць блок складається з трьох згорткових шарів із пакетною нормалізацією та активацією ReLU після кожного:

- перший згортковий шар, використовує розмір фільтра 1x1 і зменшує кількість каналів у вхідних даних. Це зменшення розмірності допомагає стиснути дані та підвищити ефективність обчислень, не жертвуючи надто великою кількістю інформації;

- другий згортковий рівень може використовувати фільтр розміром 3x3 для вилучення просторових характеристик із даних;

- третій згортковий рівень знову використовує розмір фільтра 1x1 для відновлення початкової кількості каналів перед тим, як вихід буде додано до швидкого підключення.

Як і в стандартному залишковому блоці, ключовим елементом є з'єднання швидкого доступу. Це дозволяє додавати незмінені вхідні дані

безпосередньо до виходу згорткових шарів. Це обхідне з'єднання гарантує, що суттєва інформація з попередніх рівнів зберігається та поширюється через мережу, навіть якщо згорткові рівні намагаються дізнатися додаткові функції в цьому конкретному блоці.

Завдяки поєднанню згорткових шарів для виділення ознак із швидкими з'єднаннями, які зберігають потік інформації, і введенню рівня вузьких місць для зменшення розмірності, залишкові блоки вузьких місць дозволяють ResNet-50 ефективно вирішувати проблему зникнення градієнта, тренувати глибші мережі та досягати високої точності класифікації зображень. завдання.

ResNet-50 містить 50 вузьких залишкових блоків, розташованих у стопці (рис. 2.5). Ранні рівні мережі містять звичайні згорткові та об'єднувальні шари для попередньої обробки зображення перед тим, як воно піддається подальшій обробці залишковими блоками. Зрештою, повністю пов'язані шари, розташовані на вершині структури, використовують уточнені дані для точної категоризації зображення [19].

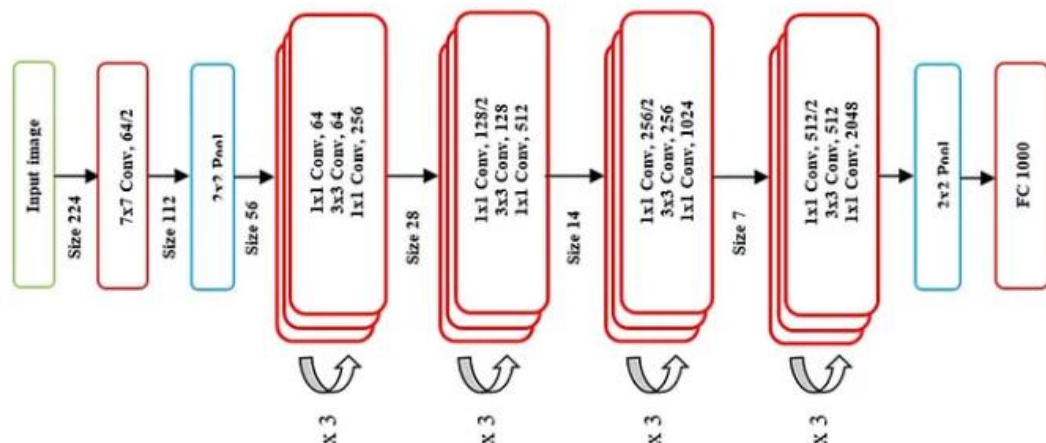


Рисунок 2.5 – Схема розташування шарів в ResNet-50 [19]

Завдяки стратегічній інтеграції вузьких залишкових блоків і коротких з'єднань ResNet-50 вміло пом'якшує проблему зникнення градієнта, дозволяючи створювати більш глибокі та потужні моделі для класифікації

зображень. Цей інноваційний архітектурний підхід відкрив двері для помітних успіхів у сфері комп'ютерного зору.

З пробуємо використати переваги цієї мережі для вирішення поставленої задачі за допомогою техніки трансферного навчання (Transfer Learning).

Трансферне навчання – це найбільш перспективна техніка для використання на стороні клієнта за допомогою TensorFlow. Найбільша перевага тут перед застосуванням тієї ж самої техніки на сервері – це збереження конфіденційності клієнтської інформації та наявність можливості доступу до сенсорів (камери, геолокації тощо).

Принцип роботи трансферного навчання полягає у наступному. Спочатку модель навчається на базі великого набору тренувальних даних. Під час процесу навчання нейронна мережа вилучає велику кількість корисних характеристик (признаків) конкретної вирішуваної задачі, які можуть бути використані як база для нової, яка буде навчатися вже на малому числі тренувальних даних для більш специфічної, але схожої задачі (рис. 2.6). Таким чином, перенавчання може відбуватися на пристроях з обмеженими ресурсами за відносно менший час.

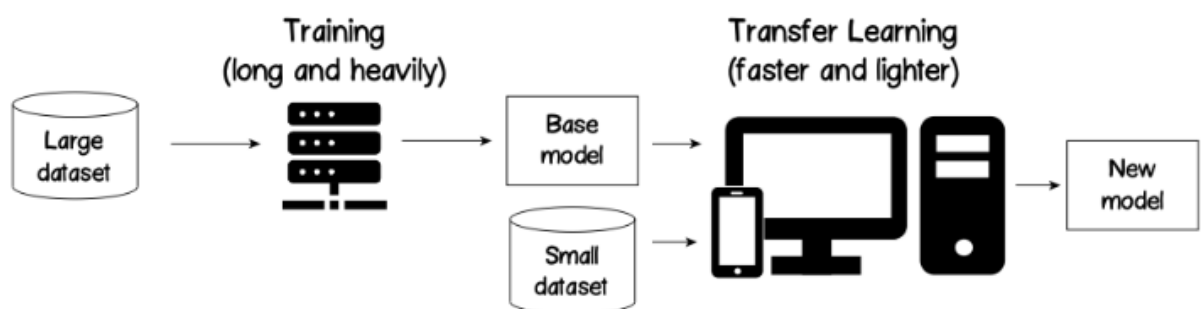


Рисунок 2.6 – Структурна схема трансферного навчання (transfer learning)

Після завантаження моделей ми можемо модифікувати та перенавчати її відповідно до вимог задачі. Так як ми будемо прагнути зменшити час на навчання моделей на базі нової вибору даних, то очевидно буде те, що необхідно «заморозити», як можна більше шарів у вихідній моделі.

Заморозити шар – це переклад усіх шарів нейронів із категорії «навчаючихся» у категорію «ненавчаючихся». Під час навчання мережі, оптимізатор буде модифікувати лише навчальні параметри мережі, що може значно скоротити час на її навчання. Це серйозно прискорює процес збільшення. Наприклад, якщо половина шарів у моделі заморожена, для тренування моделі буде потрібно вдвічі менше часу.

З іншої сторони вам все-таки потрібно навчати модель, так що якщо ми заморозимо шари занадто рано, модель буде давати неточні передбачення.

У випадку навчання моделі ResNet-50 на датасеті FER2013 були заморожені перші 170 шарів моделі (рис. 2.7).

Нейронна мережа має наступну структуру:

- `resnet50 (functional);`

Призначений для аналізу ознак, які були витягнуті базовою моделлю (ResNet50). Використання глибоких повнозв'язних шарів і Dropout забезпечує баланс між потужністю моделі та запобіганням перенавчанню. Вихідний шар softmax дозволяє класифікувати об'єкти у 7 класів (емоції).

- `flatten(input_shape=base_model.output_shape[1:]);`

Перетворює багатовимірний тензор, вихід великої моделі у вектор (одновимірний масив). Полносвязні (Dense) шари очікують, що їхній вхід буде одновимірним вектором. Це дозволяє підключити вихід згорткові частини моделі до повнозв'язних шарів.

- `dropout(0.5);`

Випадково зануляє 50% нейронів на даному шарі під час навчання. Це знижує ризик перенавчання (overfitting), що особливо важливо для моделей з великою кількістю параметрів. Під час кожної ітерації тренування випадкові нейрони тимчасово відключаються, щоб модель не залежала занадто сильно від конкретних ознак.

Model: "sequential_2"

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 2, 2, 2048)	23587712
dropout_3 (Dropout)	(None, 2, 2, 2048)	0
flatten_1 (Flatten)	(None, 8192)	0
batch_normalization_4 (Batch Normalization)	(None, 8192)	32768
dense_4 (Dense)	(None, 32)	262176
batch_normalization_5 (Batch Normalization)	(None, 32)	128
activation_3 (Activation)	(None, 32)	0
dropout_4 (Dropout)	(None, 32)	0
dense_5 (Dense)	(None, 32)	1056
batch_normalization_6 (Batch Normalization)	(None, 32)	128
activation_4 (Activation)	(None, 32)	0
dropout_5 (Dropout)	(None, 32)	0
dense_6 (Dense)	(None, 32)	1056
batch_normalization_7 (Batch Normalization)	(None, 32)	128
activation_5 (Activation)	(None, 32)	0
dense_7 (Dense)	(None, 7)	231
Total params: 23,885,383		
Trainable params: 1,335,815		
Non-trainable params: 22,549,568		

Рисунок 2.7 – Структура модифікованої ResNet-50 моделі

– dense(4096, activation='relu');

Додає повнозв'язний шар із 4096 нейронами, кожен з яких використовує функцію активації ReLU (Rectified Linear Unit). Повнозв'язні шари збирають високо абстрактні ознаки з попередніх шарів. Розмір 4096 обраний з урахуванням того, що це велика модель, яка здатна обробляти

складні ознаки. Активація ReLU прискорює навчання та допомагає уникнути проблеми зникнення градієнта.

– `dense(num_classes, activation='softmax');`

Додає вихідний шар із кількістю нейронів, рівною кількості класів (`num_classes = 7`). Використовує функцію активації `softmax`, яка нормалізує виходи нейронів у вигляді ймовірностей (сума всіх значень = 1). Цей шар визначає, до якого класу належить вхідне зображення (ймовірність для кожного класу).

Для даної мережі представимо процес її навчання в 50-ти епохах (рис.2.8).

```
Epoch 1/50
359/359 [=====] - 283s 773ms/step - loss: 1.9844 - accuracy: 0.8540 - precision: 0.2109 - recall: 0.0078 - auc: 0.5792 - f1_score: 0.0147 - val_loss: 1.8792 - val_accuracy: 0.8563 - val_precision: 0.1667 - val_recall: 0.0014 - val_auc: 0.6113 - val_f1_score: 0.0027
Epoch 2/50
359/359 [=====] - 160s 447ms/step - loss: 1.8370 - accuracy: 0.8568 - precision: 0.1990 - recall: 6.8618e-04 - auc: 0.6353 - f1_score: 0.0013 - val_loss: 1.8207 - val_accuracy: 0.8571 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00 - val_auc: 0.6430 - val_f1_score: 0.0000e+00
Epoch 3/50
359/359 [=====] - 162s 451ms/step - loss: 1.8286 - accuracy: 0.8571 - precision: 0.3802 - recall: 2.3294e-04 - auc: 0.6367 - f1_score: 4.5861e-04 - val_loss: 1.8055 - val_accuracy: 0.8571 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00 - val_auc: 0.6567 - val_f1_score: 0.0000e+00
Epoch 4/50
359/359 [=====] - 154s 430ms/step - loss: 1.8187 - accuracy: 0.8571 - precision: 0.1824 - recall: 4.2157e-05 - auc: 0.6415 - f1_score: 8.3006e-05 - val_loss: 1.8063 - val_accuracy: 0.8571 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00 - val_auc: 0.6570 - val_f1_score: 0.0000e+00
Epoch 5/50
359/359 [=====] - 156s 435ms/step - loss: 1.8107 - accuracy: 0.8571 - precision: 0.0000e+00 - recall: 0.0000e+00 - auc: 0.6453 - f1_score: 0.0000e+00 - val_loss: 1.8040 - val_accuracy: 0.8571 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
```

Рисунок 2.8 – Процес навчання модифікованої ЗНМ «ResNet-50»

У вибраній згортковій нейронній мережі застосовується функція втрат «Cross-Entropy Loss» (категоріальна перехресна ентропія або логарифмічна функція втрат) для оптимізації параметрів. Ця функція втрат використовується у задачах класифікації, де модель намагається визначити клас чи категорію вхідних даних. Основна мета функції втрат полягає в оцінці різниці між прогнозованими значеннями моделі та правильними мітками у тренувальних даних.

Функція активації `softmax` працює шляхом перетворення вхідних значень на ймовірності. Вона приймає вектор значень і повертає новий вектор, в якому кожен елемент відображає ймовірність належності до

певного класу. Таким чином, елементи знаходяться в діапазоні від 0 до 1, а їх сума рівна одиниці.

На рисунку 2.9 представлені графіки отриманні після навчання моделі:

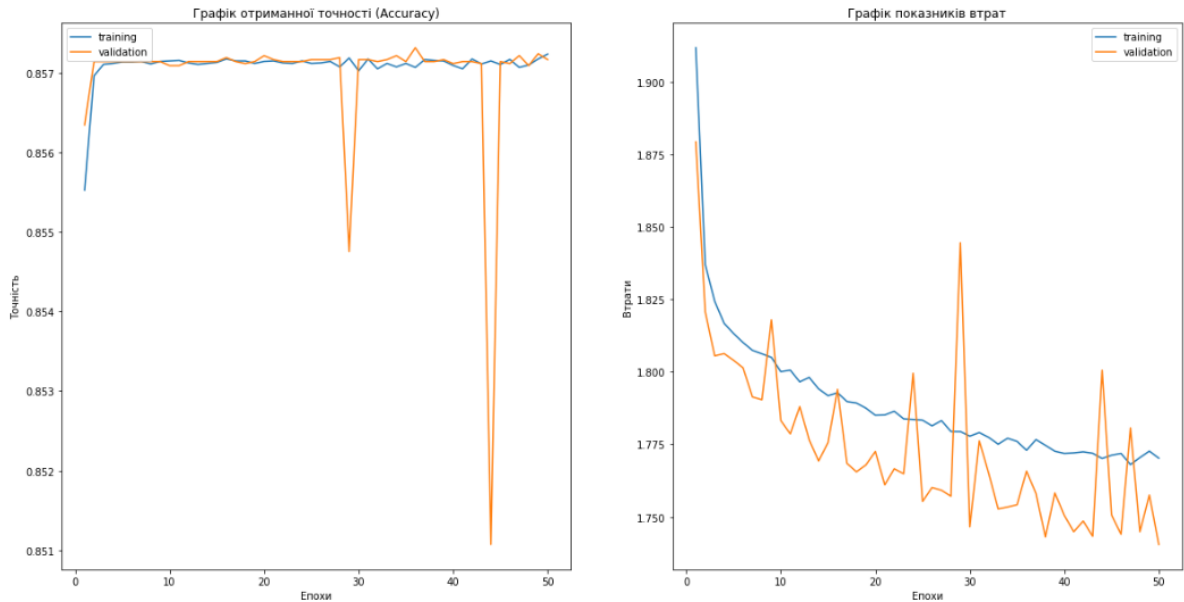


Рисунок 2.9 – Графік оцінки точності моделі та втрат

Розглянемо результати навчання, після проходження 50 циклів (епох) навчання. Дана модель має наступну матрицю невідповідностей (рис. 2.10):

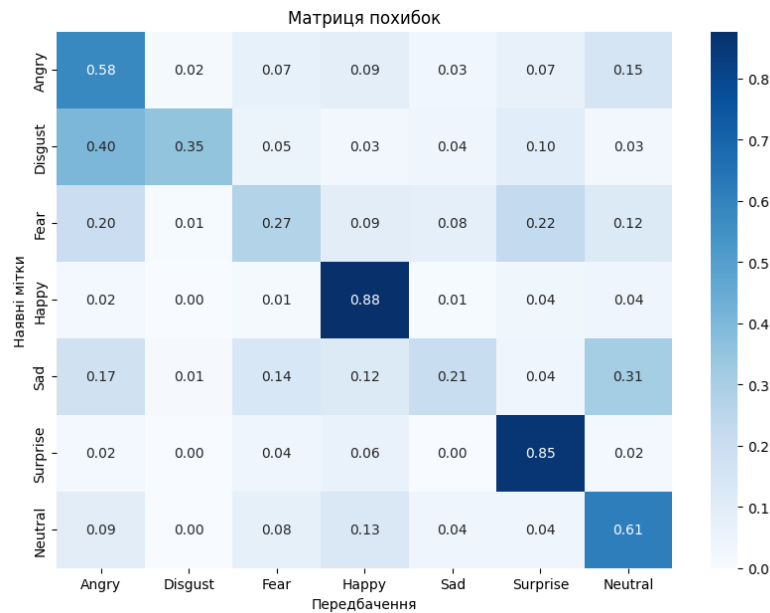


Рисунок 2.10 – Матриця невідповідностей

Нас цікавить ще один отриманий параметр, а саме розмір отриманої моделі: він становить 106,93 Мб.

Це значно кращий, порівняно з AlexNet-моделью (533 Мб), але далеко не оптимальний для використання на мобільних пристроях.

2.2.2 Згорткова мережа VGG

VGG – це 16-рівнева архітектура, деякі варіанти мали 19 рівнів. VGG має згорткові шари, шар об'єднання, ще кілька згорткових шарів, шар об'єднання, ще кілька перехідних шарів і так далі [20].

VGG базується на понятті набагато глибшої мережі з меншими фільтрами – вона використовує згортки 3×3 на всьому шляху, що є найменшим розміром фільтра conv, який дивиться лише на деякі сусідні пікселі. Він використовує маленькі фільтри через меншу кількість параметрів, що дозволяє додавати більше шарів. Він має таке ж ефективне сприйнятливое поле, як якщо б у вас був, наприклад, один згортковий шар 7×7 (рис. 2.11) [20].



Рисунок 2.11 – Приклад архітектури нейронної мережі VGGNet [21]

Ширина згорткових шарів (кількість каналів) відносно невелика: від 64 у першому шарі до 512 в останньому зі збільшенням кількості каналів у 2 рази після кожного max-pooling шару.

На жаль, мережа VGG має дві серйозні недоліки:

– дуже повільна швидкість навчання;

– сама архітектура мережі важить занадто багато. Можуть з'являтися проблеми з диском та пропускнуою спроможністю.

Через глибину та кількість пов'язаних вузлів, VGG важить більше 95Мб. Це робить процес розгортання VGG на мобільному пристрої ресурсозатратним завданням. Незважаючи на недоліки, дана архітектура є чудовим будівельним блоком для навчання, тому що її легко реалізувати.

2.2.3 Оптимізована архітектура мережі VGG

Пропонований легкий CNN натхненний мережею VGG, яку можна розглядати як спрощена мережа VGG з атрибутами кількох параметрів і низькою складністю обчислень, але з високою продуктивністю та значно меншим розміром. Ці атрибути запропонованої мережевої архітектури відповідають вимогам мобільного додатку.

Основним принципом проектування запропонованої мережевої архітектури є зменшення обчислювальна складності і параметрів мережі VGG. Насправді ми сильно скорочуємо кількість параметрів на всіх рівнях порівняно з класичною мережею VGG. Архітектура с запропонована мережа показана на рисунку 2.12 і таблиці наведеної у Додатку А.

Тому для апробації рішення нашої задачі спростимо її архітектуру, зменшивши кількість нейронів у кожному шарі і підібравши параметри навчання.

Дана згорткова нейронна мережа (CNN) складається з кількох послідовних шарів, які виконують різні операції. Її структура базується на використанні згорткових, підвибіркових (pooling), повнозв'язних шарів та шарів активації.

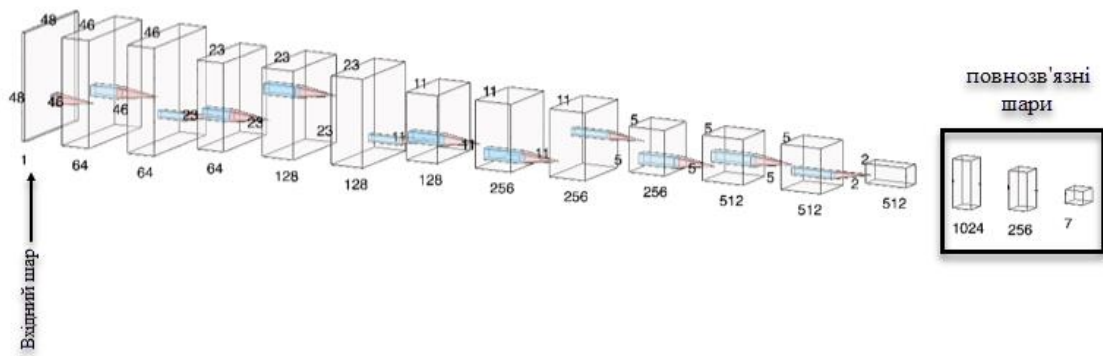


Рисунок 2.12 – Архітектура запропонованої мережі

Основні особливості архітектури:

а) вхідний шар (Input layer);

Прив'яжемо розмір першого шару до розміру зображень у датасеті. А саме встановимо 48x48. Він приймає зображення в градація сірого кольору.

б) згорткові шари (Convolutional layers);

Використовуються для виявлення ознак (features), таких як контури, текстури, і т. д. У кожному шарі фільтри (ядра згортки) мають розмір 3x3. Кількість фільтрів на кожному шарі збільшується, що дозволяє мережі витягувати складніші ознаки:

- 1) блок: 64 фільтри;
- 2) блок: 128 фільтрів;
- 3) блок: 256 фільтрів;
- 4) блок: 512 фільтрів.

Для кожного шару застосовується функція активації ReLU (Rectified Linear Unit), яка додає нелінійність і допомагає моделі виявляти складні залежності.

в) пулінг-шари (Pooling layers);

Застосовується Max Pooling із розміром вікна 2x2. Це зменшує розмір просторового представлення, залишаючи найбільш значущі ознаки. Зменшення розмірності також допомагає знизити обчислювальні витрати.

г) шар нормалізації (Batch Normalization);

Після кожного згорткового шару виконується нормалізація. Вона стабілізує навчання, зменшуючи зміщення активацій.

д) падіння (Dropout);

Після певних шарів використовується Dropout для запобігання перенавчанню. Випадково "відключаються" деякі нейрони під час навчання.

е) повнозв'язні шари (Fully connected layers);

Завершальна частина моделі складається з повнозв'язних шарів:

1) 1024 нейрони;

2) 256 нейронів;

3) вихідний шар: 7 нейронів, кожен з яких відповідає за один із класів, (категорії емоцій людини).

ж) вихідний шар (Output layer);

Функція активації: Softmax, яка переводить вихідні значення в ймовірності для кожного класу. Вона приймає вектор значень і повертає новий вектор, в якому кожен елемент відображає ймовірність належності до певного класу.

Таким чином, елементи знаходяться в діапазоні від 0 до 1, а їх сума рівна одиниці.

На рисунках 2.13 та 2.14 представлені графіки отриманні після навчання моделі:

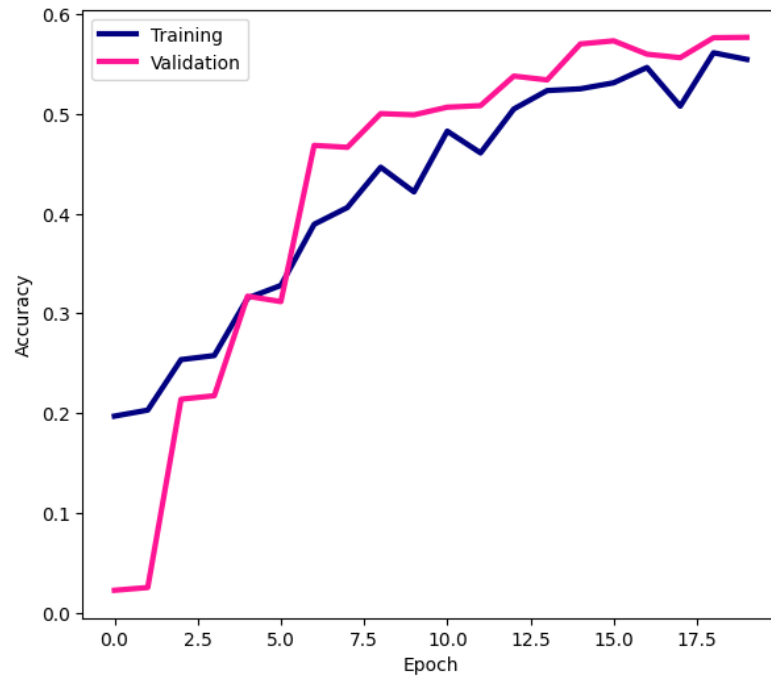


Рисунок 2.13 – Графік точності моделі

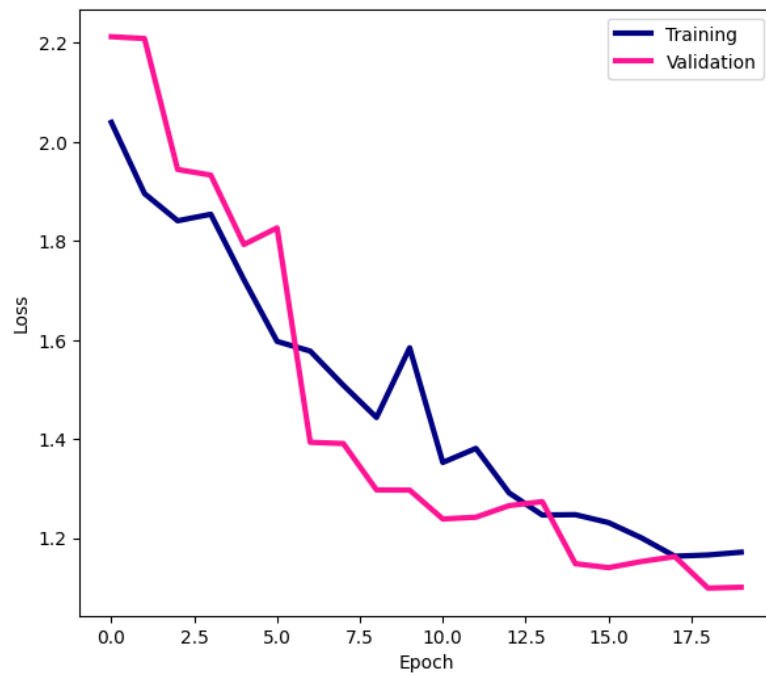


Рисунок 2.14 – Графік функції втрат

Матриця невідповідностей (рис. 2.15) показує, що по діагоналі, для всіх 7 класів емоцій ми отримуємо доволі високу точність розпізнавання, а відхілення приблизно рівномірно розповсюджені між іншими класами.

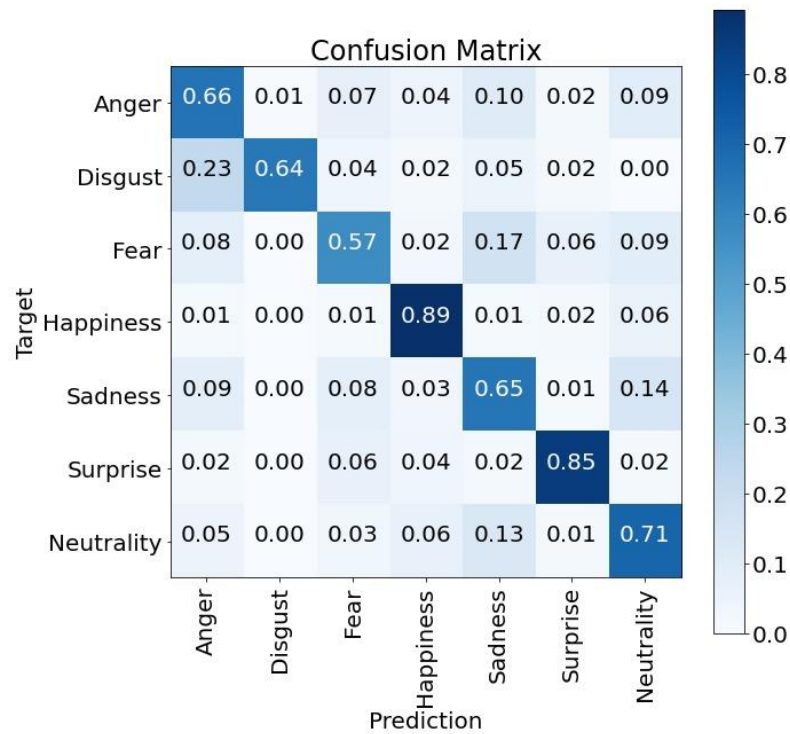


Рисунок 2.15 – Матриця невідповідностей

2.2.4 Порівняння якості побудованих моделей

В даному підпункті ми порівнюємо основні показники моделей, а саме: їх точність, повноту, матриці невідповідностей та швидкість розпізнавання. Для цього будемо використовувати дані отримані після навчання та дані, отримані в ході тестування на даних, що не брали участі у навчанні. Зведемо отримані дані у відповідну таблицю (табл. 2.1):

Таблиця 2.1 – Оцінка якості моделей

Модель	Точність, %	Повнота, %	Серед. час розпізн., с	Кількість параметрів	Розмір моделі, Мб
ResNet-50	70-75	68-73	0.428235	23 885 383	101,69
VGG-16	60-70	69-75	0,3833	14 628 679	58,51
Оптимізована VGG	65-75	58-64	0.2729	7 054 279	26,91

ResNet-50 з трансферним навчанням є ефективною моделлю для розпізнавання емоцій, оскільки її архітектура дозволяє зберігати високу продуктивність навіть на обмежених даних.

Але її розмір значно більше, ніж планується отримати. Що робить її не дуже підходящою для використання на мобільних пристроях.

Альтернатива у вигляді моделі збудованій на архітектурі VGG-16 виглядає більш привабливою для вирішення поставлених задач. Вона має прийнятні показники точності, що робить VGG-16 доволі потужною. Але її розмір ще можна зменшити за рахунок зменшення кількості параметрів на першому вхідному шарі. Тому що для навчання ми використовуємо датасет «FER2013», де зображення мають маленький розмір та представлені у градації сірого кольору.

Таким чином, з точки зору точності, розмірів та середньої швидкості розпізнавання найкращими виявилась запропонована оптимізаційна модель на архітектурі VGG. Саме її будемо застосовувати при розробці мобільного додатку на платформі iOS.

2.3 Висновки до розділу 2

У цьому розділі досліджується оцінка та порівняння різних моделей для класифікації та розпізнавання емоцій. Проведене порівняння якості моделей дозволяє стверджувати, що для оптимізованої архітектури CNN, такі як вдосконалена VGG, є найбільш ефективними.

Залежно від вимог до ресурсів, використання більш легких або спеціалізованих архітектур може забезпечити баланс між швидкістю та точністю, що є критичним для реальних застосувань у мобільних пристроях чи інших обмежених середовищах.

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ РОЗПІЗНАВАННЯ ЕМОЦІЙ

Процес розробки мобільного додатку з інтеграцією машинного навчання вимагає врахування як технічних, так і практичних аспектів, зокрема вибору інструментів для інтеграції, управління камерами пристроїв та реалізації ефективних алгоритмів розпізнавання. Тому в роботі особливу увагу приділено вибору інструментів розробки, зокрема Xcode для iOS, та оптимізації моделей машинного навчання під специфічні вимоги мобільних пристроїв.

3.1 Вимоги для розробки мобільного додатку

Для багатьох мобільних додатків загальні рішення для інтеграції зі штучним інтелектом можна грубо розділити на дві категорії: хмарне рішення та локальне рішення.

Стратегія локального рішення дозволяє навчити фіксовану модель глибокого навчання, таку як CNN, навчати за допомогою високопродуктивного комп'ютера, а потім розгорнути це рішення на мобільних пристроях або в хмарі.

Хмарне рішення дозволяє завантажувати дані в хмару, а потім отримати результати, що може бути найкращим рішенням для багатьох мобільних програм.

Це тому, що ми не потрібно враховувати обмежені ресурси мобільних пристроїв. Ми можемо заощадити місце для зберігання і використовуємо кластер високопродуктивного процесора або графічного процесора з хмари для підвищення швидкості алгоритму виконання. Однак для афективного обчислення обличчя хмарне рішення не є ідеальним рішенням.

У нього є два недоліки, а саме:

- проблема конфіденційності;
- затримка мережі.

Оскільки хмарне рішення потребує завантажувати й отримувати дані з хмари, мережеве середовище буде важливим для безперебійної роботи і продуктивності. Будь-яке випадання даних і затримка спричиняють незручності під час обміну даними.

Тому локальне розташування є альтернативним рішенням і більше підходить і яке могло б уникнути вищезгаданих проблем. Однак через обмеженість ресурсів на мобільних пристроях, треба враховувати складність архітектуру CNN.

Тобто, треба шукати компроміс між апаратними обмеженнями, такі як обчислювальні ресурси або простір для зберігання, і продуктивністю моделей CNN.

У цій роботі, відповідно до вимог мобільної розробки, ми розробляємо архітектуру CNN з метою пошуку цього компромісу.

Перелік вимог які треба обов'язково виконанати при розробці мобільного додатку на платформі iOS:

- у вас повинен бути встановлений Xcode;
- ви повинні мати дійсний ідентифікатор розробника Apple;
- програма потребує камери, і її потрібно запустити на справжньому пристрої iOS. Ви можете створити його та запустити за допомогою симулятора iPhone, але додаток викликає виключення камери не знайдено;
- інтеграція навченої моделі у мобільний додаток за допомогою інструментів Apple CoreML Framework.

Також знадобляться інструменти командного рядка Xcode: `xcode-select --install` Якщо це нова інсталяція, вам потрібно буде запустити програму Xcode один раз, щоб погодитися з ліцензією, перш ніж продовжити.

3.2 Вибір інструментів розробки програми

Виходячі з визначених вимог до мобільного додатку зробимо вибір мови програмування та среди розробки (табл. 3.1):

Таблиця 3.1 – Порівняння мов програмування та інструментів розробки

Мова програмування	Середовище розробки	Ліцензія	Підтримка роботи з ML	Підтримка роботи з камерою
Swift	Xcode	Відкрита	Повна підтримка через Core ML	Повна підтримка
Objective-C	Xcode	Власна (Apple Proprietary)	Обмежена підтримка через Core ML	Повна підтримка
Python	Pyto, Juno	Відкрита	Обмежена підтримка через бібліотеки Core ML, TensorFlow для iOS	Обмежена підтримка через бібліотеки, такі як OpenCV
C# (Xamarin)	Xamarin Studio	Відкрита	Підтримка через ML.NET,	Підтримка через Xamarin.iOS
C++	Xcode, CLion	Відкрита	Обмежена підтримка через Core ML	Підтримка через AVFoundation
JavaScript (React Native)	Visual Studio Code	Відкрита	Обмежена підтримка через бібліотеки	Підтримка через React Native Camera API
Dart (Flutter)	Visual Studio Code	Відкрита	Підтримка через сторонні бібліотеки для ML	Підтримка через Camera plugin в Flutter

Із приведенної таблиці легко визначаємо, що оптимальним інструментом для розробки є XCode, а мовою програмування є Swift. Обидві системи засновані і розробляються корпорацією Apple, є безкоштовними і доступні на платформі macOS.

Xcode – це програма для macOS, створена компанією Apple для розробки додатків. Це єдиний офіційний інструментарій для розробки додатків для iOS чи інших ОС Apple (рис. 3.1).

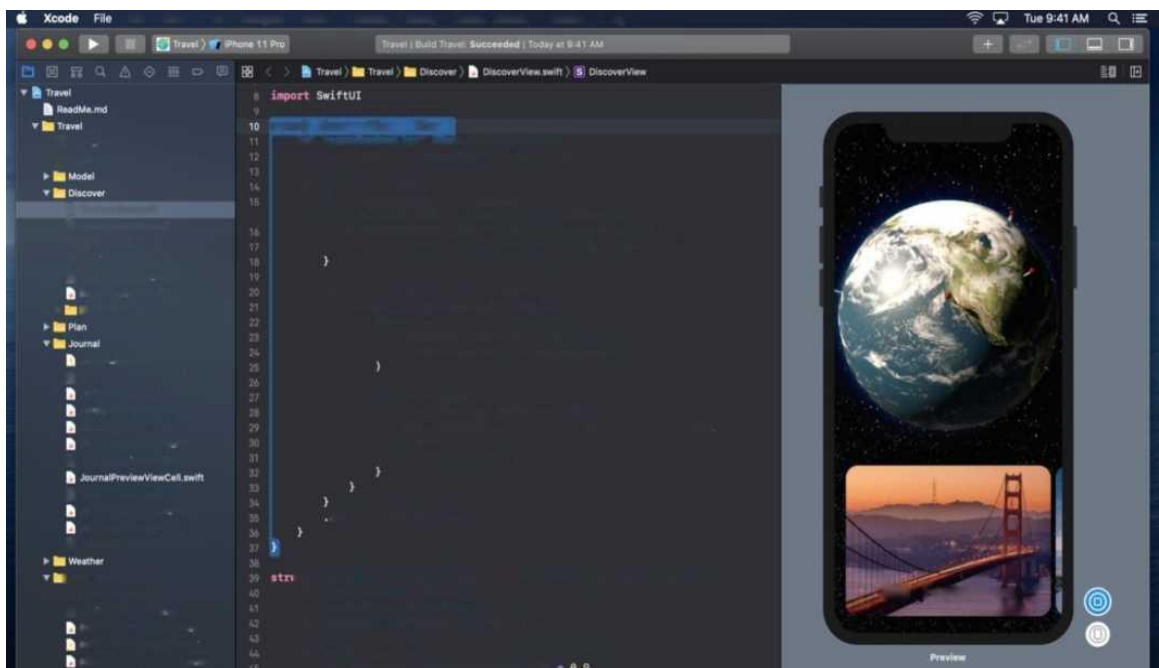


Рисунок 3.1 – Вигляд Xcode [22]

Він пропонує безліч різних функцій і переваг для розробки додатків iOS та включає в себе інструменти, які допомагають розробнику на кожному етапі інженерного процесу. Дана програма дає можливість створювати застосунки, які у подальшому могли б бути опубліковані у магазині додатків Apple.

Xcode включає в себе всі інструменти: текстовий редактор, компілятор і систему збірки. За допомогою Xcode можна писати, компілювати і налагоджувати додаток, а після закінчення, можна відправити його в Apple App Store.

Xcode у світі програмування характеризується відмінними інструментами налагодження, які дозволяють розробникам швидше вирішувати проблеми в своєму додатку. Також він є корисним при організації управління ресурсами зображень і файлів коду завдяки вбудованим інструментам (рис. 3.2).

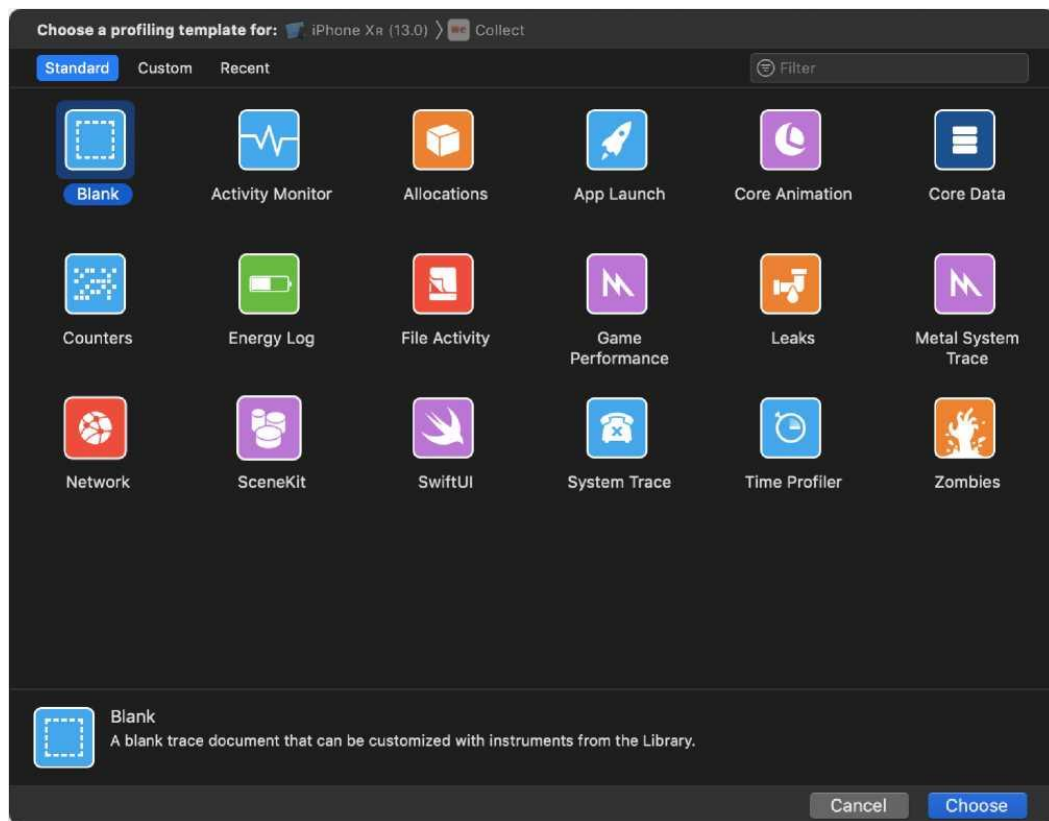


Рисунок 3.2 – Панель інструментів [23]

3.3 Основні рішення щодо реалізації класів

Застосунок повинен базуватись на архітектурі «MVP» та основних принципах реалізації мобільних додатків для платформи iOS. Також мобільний додаток має містити в собі залежності від основних фреймворків для інтеграції моделей машинного навчання, які були обрані для даного дослідження.

На рисунку 3.3 наведено діаграму класів мобільного додатку. На діаграмі відображено усі основні компоненти, що містять логіку відображення екранів і реалізацію задачі розпізнавання образів.

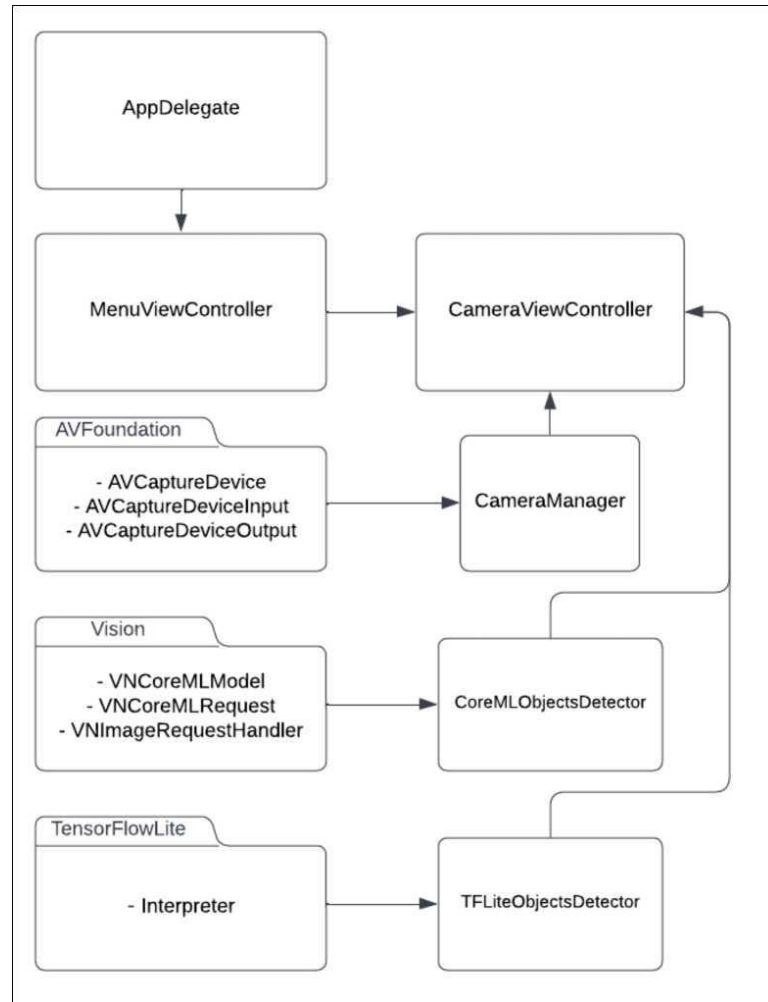


Рисунок 3.3 – Діаграма класів мобільного додатку

Відправною точкою для старту роботи мобільного додатку є клас «AppDelegate». Саме цей клас є основним керуючим компонентом та відповідає за навігацію в середині додатку. Основною його функціональною відповідальністю є запуск першого основного екрану, який надасть користувачу доступ до функціоналу мобільного додатку.

Наступним важливим компонентом розробленого мобільного додатку є клас «CameraViewController». Основна задача даного класу полягає в реалізації саме користувацького інтерфейсу. Після того, як налаштування

камери завершено, відбувається перехід на наступний екран, а саме «CameraViewController». Що стосується даного компоненту, то його основна функціональна відповідальність полягає в наданні користувачу доступу до даних з камери та відображення результатів розпізнавання отриманих кадрів у вигляді підсвічування знайдених моделлю об'єктів.

Що стосується класів з приставкою «Detector», то вони є свого роду обгорткою для моделей розпізнавання образів, які використовуються мобільним додатком. В середині себе вони інкапсулюють логіку ініціалізації моделей та виконання задач розпізнавання на відповідних зображеннях. Логіка повинна складатись з наступних етапів:

- первинна обробка вхідного зображення, яка полягає у зміні формату до такого, який необхідний моделі на вхід;
- зміна розмірів вхідного зображення до розміру, з яким працює інтегрована модель машинного навчання;
- обробка результатів отриманих від моделі після розпізнавання.

В результаті виконання усіх вищезазначених етапів класами-обгортками усі отримані дані мають бути передані до класу «CameraViewController». Даний клас має виконати певні перетворення, для того щоб правильно відобразити отримані результати. Відображення результатів має полягати у підсвіченні знайденого об'єкта за допомогою кольоровою рамки з відповідною назвою.

Класи-обгортки використовують відповідні фреймворки, які дають змогу взаємодіяти з моделями машинного навчання, що використовуються. Для «CoreMLObjectsDetector» допоміжною бібліотекою виступає «Vision» (побудований на базі CoreML).

Для того, щоб отримати контроль над камерою девайсу і мати змогу отримувати кадри для подальшого аналізу за допомогою моделей використовується клас «CameraManager». Даний клас використовує фреймворк «AVFoundation», який є частиною iOS SDK. Ця бібліотека дає доступ розробникам до камер відповідного девайсу. Основною

функціональною задачею даного класу є надання доступу до кадрів з камер девайсу, які виступають вхідними параметрами для моделей машинного навчання для подальшого розпізнавання образів.

Взаємодія між цими класами фактично реалізує інтеграцію мобільного додатку з відповідними моделями машинного навчання через класи обгортки з використанням камери девайсу як джерела вхідних даних для розпізнавання об'єктів.

3.4 Інтеграція з Apple CoreML

На WWDC'17 Apple представила новий фреймворк для роботи із технологіями машинного навчання Core ML. На основі нього в iOS реалізовані власні продукти Apple: Siri, Camera та QuickType. Core ML дозволяє спростити інтеграцію машинного навчання у додатки та створювати різні «розумні» функції. Це дозволяє створювати інтелектуальні фічі на пристрої, в тому числі й розпізнавання будь-яких об'єктів. Core ML спрощує інтеграцію різних алгоритмів машинного навчання у додатки. Результати обчислень відбуваються майже миттєво [24].

Фреймворк Core ML використовує низькорівневі фреймворки Metal, Accelerate та BNNS, що забезпечує високу швидкість обчислень (рис. 3.4).

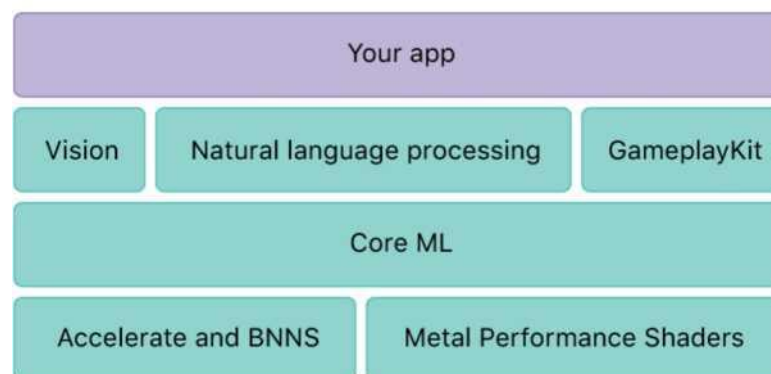


Рисунок 3.4 – Схема роботи CoreML в мобільному додатку [24]

Ядро підтримує нейронні мережі, узагальнені лінійні моделі, деревоподібні алгоритми прийняття рішень, генерацію ознак, метод опорних векторів, пайплайн моделі[24].

Однією з ключових переваг CoreML є простота використання. Фреймворк надає простий API для завантаження та використання моделей машинного навчання. CoreML підтримує широкий спектр моделей машинного навчання, включаючи нейронні мережі, ансамблі дерев і опорні векторні машини. Ці моделі можна навчити за допомогою різноманітних інструментів і мов, наприклад Python, а потім експортувати у формат, сумісний із CoreML.

Ще однією ключовою особливістю CoreML є його ефективність. Фреймворк оптимізований для мобільних пристроїв, що означає, що він розроблений для ефективної роботи на пристроях iOS при мінімальному споживанні енергії.

Незважаючи на велику кількість позитивних аспектів використання технології CoreML, в ній є також свої особливості та обмеження. Моделі інтегровані у додатки не шифруються. Розробникам доведеться подбати про захист даних самому.

Незважаючи на це саме ця технологія була обрана для дослідження використання моделей машинного навчання для задач розпізнавання емоцій, адже вона є нативною і максимально оптимізованою для використання на девайсах з операційною системою iOS.

3.5 Адаптування моделі до використання у додатку

У TensorFlow Lite вбудовані механізми які допомагають перетворити, попередньо підготовлену модель, яка знаходиться в TensorFlow, у конкретний формат, який підходить для портативних пристроїв, враховуючи потребу оптимізації швидкості та зберігання. Цей формат добре працює для мобільних пристроїв.

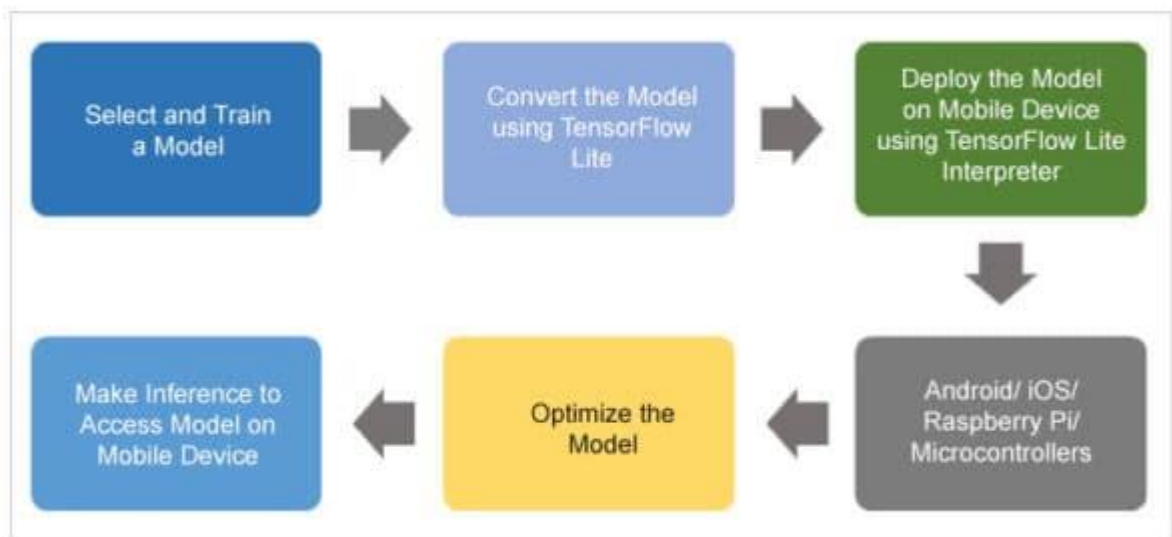


Рисунок 3.5 – Робочий процес Tensorflow Lite [25]

Етапи робочого процесу:

- а) завантажити попередньо підготовлену модель;
- б) перетворити навчену модель у версію Tensorflow Lite для мобільних пристроїв (рис. 3.6).

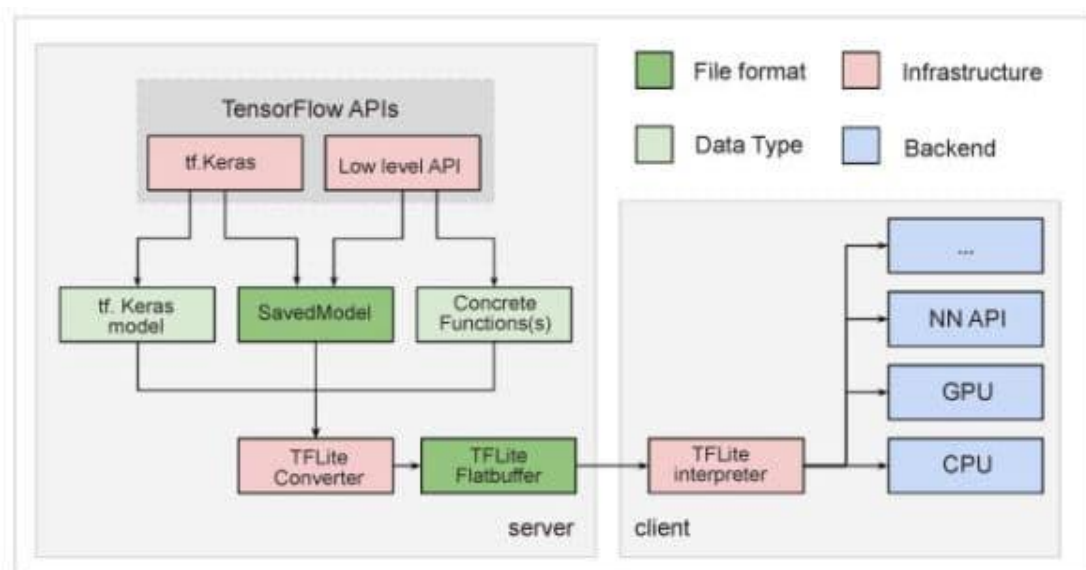


Рисунок 3.6 – процес перетворення в Tensorflow Lite [25]

Використовуючи перетворювач TensorFlow, модель TensorFlow перетворюється у файл плоского буфера, який називається .tflite-файл, який розгортається на мобільні або вбудовані пристрої (лістинг 3.1).

Лістинг 3.1:

```
# Імпорт бібліотеки Core ML Tools та TensorFlow
import coremltools
import tensorflow as tf

# Збереження моделі після процесу навчання на диск у форматі HDF5 (файл з
розширенням .h5).
# Цей формат містить архітектуру моделі, ваги, оптимізатор та стан навчання.
model.save('model.h5')

# модель завантажується назад у змінну keras_model
keras_model= tf.keras.models.load_model('model.h5')

# конвертація моделей у формат TensorFlow
converter = tf.lite.TFLiteConverter.from_keras_model(keras_model)

# модель конвертується у формат TensorFlow Lite,
# який є компактнішим та оптимізованим для виконання на пристроях із обмеженими
ресурсами, таких як смартфони,
tflite_model = converter.convert()
```

В одному файлі зберігається інформація пов'язана з вагами, зміщенням та конфігурацією тренувань. Спільний доступ та розгортання моделі стає легкою із збереженою моделлю.

Наступний код використовується для перетворення збереженої моделі в CoreML-формат для того, щоб можна її використати у iOS додатку (лістинг 3.2).

Лістинг 3.2:

```
# Імпорт бібліотеки Core ML Tools та TensorFlow
import coremltools
import tensorflow as tf

# Шлях до моделі
model_path = 'model.h5'

# Завантаження моделі Keras: завантажує модель з файлу model.h5,
# яка була раніше навчена і збережена.
keras_model = tf.keras.models.load_model(model_path)

# перетворює модель Keras у формат Core ML
model = coremltools.convert(keras_model, convert_to="mlprogram")

# зберігає конвертовану модель у файл model.mlcore,
```

```
# який можна використовувати в додатках iOS/macOS  
model.save("model.mlcore")
```

3.6 Висновки до розділу 3

Загалом, використання сучасних фреймворків, таких як Core ML та TensorFlow Lite, дозволяє створювати потужні мобільні додатки з інтелектуальними функціями на базі машинного навчання, що можуть використовуватися у реальних умовах з високими вимогами до ефективності та продуктивності.

4 ЕКСПУЛАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Експлуатація програми

В результаті реалізації усіх основних складових частин мобільного додатку ми отримали систему взаємодій, які разом представляють собою кінцевий застосунок. Однак, ключовим аспектом з точки зору поставленої задачі, а саме інтеграції моделі розпізнавання образів в мобільний додаток, є ефективність. Саме для підвищення даної характеристики було реалізовано певний алгоритм, який сприяє більш інтелектуальному аналізу вхідних даних моделю та безпосередньо впливає на кінцеву якість додатку.

Оскільки для розпізнавання використано камеру, то вхідною умовою для даного додатку є доступ до усіх кадрів, які отримуються пристроєм в режимі реального часу. У зв'язку з розвитком технологій та безпосередньо вдосконаленням мобільних пристроїв на сьогоднішній день користувачі мають можливість знімати відео дуже високої якості зі швидкістю від 30 до 120 кадрів за секунду в межах можливостей смартфонів на базі системи iOS. Таким чином, ми отримуємо, що у випадку прямої реалізації задачі розпізнавання образів ми стикнемося з тим, що моделі буде отримувати мінімум 30 кадрів в секунду на обробку. В такому випадку ресурси будуть використовуватись на максимум і не будуть віддавати точні результати.

Значить, для більш ефективної реалізації необхідно зробити так, щоб не всі кадри потрапляли на розпізнавання. На рисунку 4.1 зображено блок-схема для алгоритму розпізнавання образів за допомогою CoreML моделі машинного навчання в мобільному додатку.



Рисунок 4.1 – Алгоритм розпізнавання за допомогою CoreML моделі

Як ми можемо бачити на вищезазначеній діаграмі, алгоритм включає в себе додатковий підрахунок кадрів для більш оптимального розпізнавання.

Після отримання кадру з камери ми збільшуємо лічильник на одиницю. Якщо даний кадр є 15-м, то ми відправляємо його на обробку до моделі. Після отримання результатів ми відображаємо їх на екрані. Обробка кожного 15-го кадру підвищить продуктивність системи й знизить навантаження на апаратне забезпечення пристрою.

4.2 Методика та результати тестування

Основними функціями програми є :

- виявлення обличчя у відеопотоці з камери в реальному часі;
- відображення обличчя на екрані з відповідними рамками, категоріями (емоціями) та ймовірностями.

При першому запуску програми на екран буде виведено повідомлення з питанням про допуск до вбудованій камері пристрою(рис. 4.2).

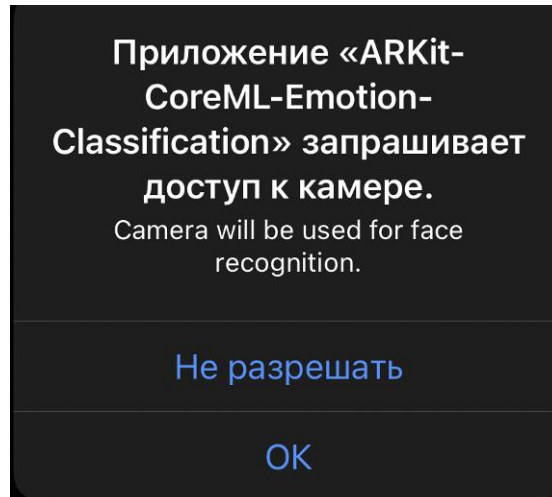


Рисунок 4.2 – Повідомлення про запит на доступ до камери

Наведемо приклади розпізнавання емоцій розпізнаний при роботі мережі на iPhone X:

а) експеремент з обличчям під кутом (рис. 4.3);



Рисунок 4.3 – Розпізнавання емоцій, випадок з поворотом голови більше 15 градусів

Як бачимо модель встигає обробляти кадри та показує високий відсоток розпізнавання, якщо не робити різких рухів головою.

б) експеримент з різним положенням (рис. 4.4).

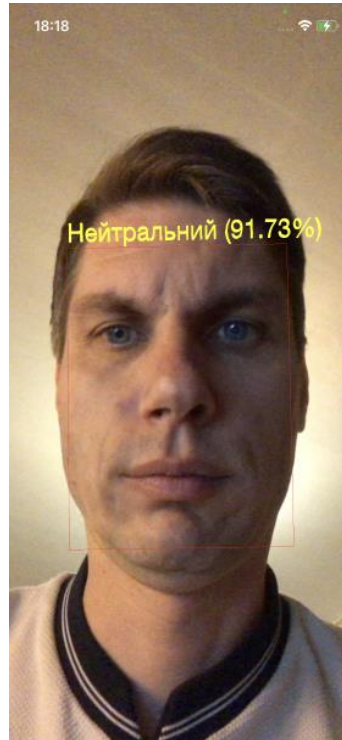


Рисунок 4.4 – Розпізнавання емоцій, дивлячись прямо у камеру

Експеримент показує, що якщо ми міняємо місто розположення джерала освітлення, то це не дуже сильно впливає на точність отриманих результатів.

Спробуємо зробити експеримент при поганому освітленні і на відстані близько метра (рис. 4.5).

Як бачемо, отримані результати набагато гірші за попередні, і дають невірні результати. Це пояснюється обмеженим розміром датасета на якому навчалася модель та відсутності в ньому подібних випадків. Тобто, модель не була навчена на таких зображеннях і потребує донавчання.

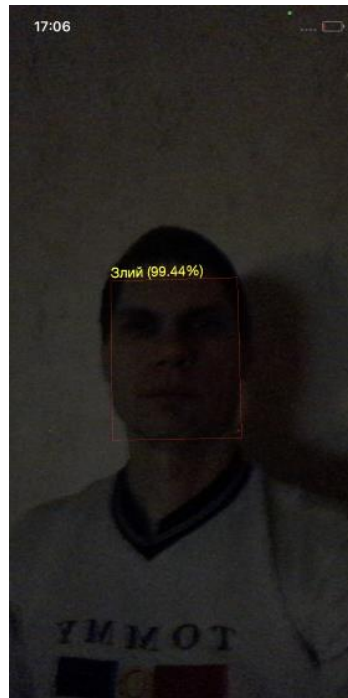


Рисунок 4.5 – Розпізнавання емоцій, з плохим рівнем освітлення і дальній дистанції

Зробимо детальний огляд самих популярних датасетів, на яких навчаються моделі розпізнавання емоцій людини.

4.3 Огляд та порівняння датасетів для розпізнавання емоцій

4.3.1 Датасет AffectNet

Набір даних AffectNet є колекцією зображень обличчя, які було отримано з реальних умов та анотовано вручну. До складу цього датасету входить понад мільйон фотографій облич, зібраних з Інтернету шляхом запитів у популярних пошукових системах. При цьому використовувались ключові слова, пов'язані з емоціями, подані на шести різних мовах.

Приблизно половина зібраних зображень (близько 440 тисяч) була анотована вручну для виявлення семи дискретних виразів обличчя (за категоріальною моделлю) та для вимірювання інтенсивності валентності та збудження (за розмірною моделлю). AffectNet вважається найбільшою базою даних виразів обличчя, валентності та збудження, зібраною у реальних

умовах, що надає можливість проведення досліджень щодо автоматичного розпізнавання виразів обличчя за допомогою різних моделей емоцій [26].

Наразі тестовий набір даних ще не був випущений. У найближчому майбутньому розробники цього набору даних планують організувати виклик на AffectNet, і тестовий набір буде використаний для оцінки методів та алгоритмів учасників.

4.3.2 Датасет EMOTIC

Анотації в наборі даних EMOTIC (рис. 4.4) об'єднують два методи представлення емоцій: дискретні категорії та безперервні виміри.



Рисунок 4.4 – Приклади розмітки датасету EMOTIC [27]

Цей набір даних також містить анотації, створені на основі реальних життєвих ситуацій. Поєднує два методи представлення емоцій: дискретний і континуальний. Містить анотації для реальних життєвих ситуацій.

У ньому визначено 26 дискретних категорій (наприклад, радість, сум, страх, гнів тощо) та 3 безперервні виміри: валентність, збудження, домінування.

В датасеті є 26 різних категорій, які дозволять детально моделювати емоції, враховуючи контекст ситуацій.

Але при цьому, велика кількість категорій може ускладнити навчання моделей. Також можливі суб'єктивні помилки при ручній анотації.

4.3.3 Датасет FER2013

База даних, використана в змаганні «Визначення мімічних виразів обличчя», що відбулася в Kaggle у 2013 році (FER 2013), складається із 35,887 зображень обличчя із сімома класами виразів обличчя [28].

Дані складаються з 48x48 піксельних зображень сірого кольору. Обличчя автоматично зареєстровані так, що обличчя було більш -менш зосередженим і займає приблизно однакову кількість місця на кожному зображенні. Завдання полягає в тому, щоб класифікувати кожне обличчя на основі емоції, показаної в виразі обличчя, в одній із семи категорій.

Якщо структура нейронної мережі розроблена для обробки зображень із низькою роздільною здатністю, то вхідні зображення з високою роздільною здатністю необхідно масштабувати відповідно до вимог цієї структури. У процесі масштабування може відбуватися спотворення пропорцій зображення та виникати розмиття, що призводить до зниження продуктивності класифікації. Крім того, некоректна анотація даних погіршує результати класифікації. На рисунку 4.5 наведено приклади зображень із бази даних FER2013, які були неправильно анотовані для різних виразів обличчя.



Рисунок 4.5 – Приклади зображень кожного з класів у «FER2013» [29]

Слід зазначити, що класи в наборі даних незбалансовані: класі «радість» зберігаються 8989 фотографій, тоді як клас "огида" складається лише з 547 фотографій. На категорії «гнів», «страх», «сум», «здивування» та «нейтральний вираз обличчя» припадає 6198, 5121, 6077, 4002 та 6198 фотографій відповідно (рис. 4.6).

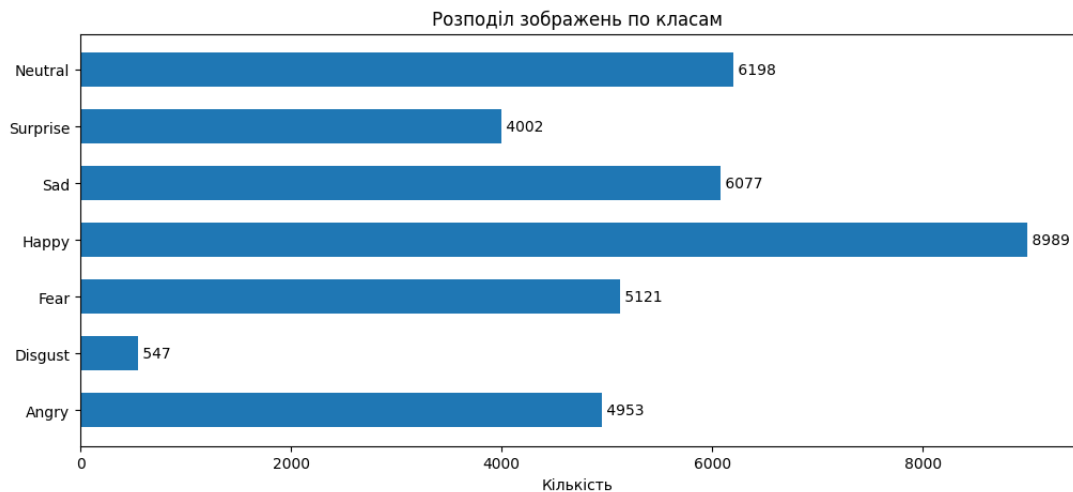


Рисунок 4.6 – Розподіл кількості зображень по класам в «FER2013» [30]

4.3.4 Порівняння датасетів

Як видно, існує велика кількість різноманітних баз даних. Основними викликами при їх створенні є кілька аспектів. По-перше, це складність розмітки даних, яка зазвичай ґрунтується на експертних оцінках, що вимагає високого рівня професійності й значних ресурсів. По-друге, складність забезпечення природності даних. Якщо учасника попередньо попросити зобразити певну емоцію, виникає питання автентичності отриманого виразу. Якщо ж людину не попереджати, це створює етичні та технічні труднощі, зокрема, як організувати процес так, щоб учасник не здогадався про зйомку, але при цьому забезпечити якість отриманих даних. Порівняємо між собою наведені вище датасети (табл. 4.1).

Таблиця 4.1 – Порівняльна таблиця

Характеристика	AffectNet	EMOTIC	FER2013
Обсяг даних	1+ мільйон (440k ановані)	34,000+	35,887
Кількість категорій	7 +	26 +	7

Продовження таблиці 4.1

Характеристика	AffectNet	EMOTIC	FER2013
Якість зображень	Висока, реальні умови	Висока, реальні умови	Низька (48х48 пікселів)
Тип анотації	Ручна	Ручна, змішаний підхід	Ручна

ВИСНОВКИ

Результати, отримані в роботі, є рішенням практичної задачі підвищення ефективності прогнозування емоцій людини в реальному часі. Побудувавши кілька часто зустрічаються у завданнях класифікації моделей можна зробити висновок про їх ефективність в рамках роботи з розпізнавання емоцій людини за допомогою комп'ютерного зору.

Проведене дослідження продемонструвало, що завдання розпізнавання емоцій людини можна вирішити кількома сучасними методами, кожен з яких здатний показати вартий уваги результат.

Високефективність може мати як проста згорткова нейронна мережа, що складається лише з декількох шарів, так і складні до навчені моделі, наприклад ResNet50, побудовані за правилами навчання, що переноситься.

Основні складнощі, що виникають під час навчання моделей – незбалансованість класів у вихідному наборі даних, а також проблема перенавчання нейронних мереж.

Шляхом збільшення кількості зображень у наборі даних можна позбутися від наявного дисбалансу в наборі «FER2013» і, як результат, усунути складність у розпізнаванні емоцій, зразків яких недостатньо для якісного навчання нейронної мережі. З перенавчанням моделей можна боротися, застосовуючи техніку «проріджування» нейронів – Dropout.

Завдяки наявним бібліотекам, таким як Apple Core ML та TensorFlow, стало можливим впровадження нейронних мереж безпосередньо на мобільні пристрої замість використання хмарних сервісів. Незважаючи на те, що вимоги до обчислювальних ресурсів і розміру мереж, доволі високі, вони пропонують набори інструментів, які надають корисні нові можливості в програмах смартфонів. І є багато переваг у виконанні машинного навчання локально, а не в хмарі, включаючи меншу затримку, більшу доступність, покращену конфіденційність та менші витрати.

Розглянуті моделі нейронних мереж можуть бути додатково доопрацьовані та використані як модуль розпізнавання емоцій у програмних продуктів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Іванюк В. І., Потапова К. Р., Наливайчук М. В. Огляд нейронних мереж та стандартних алгоритмів обробки зображень в області комп'ютерного зору : навчальний посібник / Міжнародний науковий журнал: Інтернаука №15, 2023. — 211 с.
2. Facial recognition systems [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Facial_recognition_system.
3. Vartanov A. Facial expressions and subjective assessments of emotions. / Cognitive Systems Research. 2020, p. 319-328.
4. Computer Vision: What it is and why it matters. [Electronic resource]. – Access mode: https://www.sas.com/en_us/insights/analytics/computer-vision.html.
5. Устенко І.В., Назаренко В.О. Удосконалення математичної моделі Віюлі-Джонса для розпізнавання обличчя у відеопотоці та розробка програми для її реалізації / Вчені записи ТНУ ім. В.І. Вернадського: серія технічні науки. – Херсон, том 29(68) Ч.2 №5, 2018 – 174 с.
6. А.К. Изтай, Ж.Р. Уалиев, А.О. Кабдолдина. Исследование алгоритмов распознавания и идентификации при видеофиксации в реальном времени / Алматинский технологический университет, Алматы, Казахстан. №1 (57), 2024 – 53 с.
7. Психологія, філософія і роздуми про життя. [Electronic resource]. – Access mode: <https://ua.sainte-anastasie.org/articles/bienestar/cuntas-emociones-existen.html>.
8. Financial Times. [Electronic resource]. – Access mode: <https://www.ft.com/womenof2023>.
9. О. О. Сліпченко, С. В. Клименко. Аналіз сучасного стану інформаційних технологій оцінки психоемоційного стану людини. Актуальні проблеми автоматизації та інформаційних технологій / Актуальні проблеми

автоматизації та інформаційних технологій. Том 23. ISSN 2312-119X, 2019 – 52-58 с.

10. Marc Chaumont. Digital Media Steganography: Principles, Algorithms, Advances / M. Hassaballah, Montpellier University, France: 2019 – 386p.

11. Convolutional Neural Networks (CNN): Summary. [Electronic resource]. – Access mode: <https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-summary>.

12. FiceReader. [Electronic resource]. – Access mode: <http://www.noldus.com/facereader>.

13. EmoDetect. [Electronic resource]. – Access mode: <http://www.yeschat.ai/ru/gpts-2OToEnXLAg-EmoDetect>.

14. the FacE Analysis System. [Electronic resource]. – Access mode: <http://mmer-systems.eu/>.

15. Dornaika, F.; Raducanu, B. Efficient facial expression recognition for human robot interaction. / In Proceedings of the 9th International Work-Conference on Artificial Neural Networks on Computational and Ambient Intelligence. San Sebastián: Spain, 2007. – 700–708 p.

16. Facial point annotations [Electronic resource]. – Access mode: <https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/>.

17. Methods for Facial Expression Recognition with Applications in Challenging Situations. [Electronic resource]. – Access mode: <https://onlinelibrary.wiley.com/doi/10.1155/2022/9261438>.

18. Kaiming He, Xiangyu Zhang, Shaoqing Ren. Deep Residual Learning for Image Recognition / Microsoft Research. [Electronic resource]. – Access mode: <https://arxiv.org/pdf/1512.03385>, 2015 - 12p.

19. Bendjillali R. I. Illumination-robust face recognition based on deep convolutional neural networks architectures / Indonesian Journal of Electrical Engineering and Computer Science Vol 18(No 2), 2020 - 1015~1027pp.

20. VGGNet / Wikipedia [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/VGGNet>.
21. Vgg 16 Architecture, Implementation and Practical Use / Medium, 2022. [Electronic resource]. – Access mode: <https://medium.com/pythoneers/vgg-16-architecture-implementation-and-practical-use-e0fef1d14557>.
22. Apple Xcode. [Electronic resource]. – Access mode: https://www.reddit.com/r/iOSProgramming/comments/foa2y4/xcode_114_is_out_now/?rdt=48506.
23. Xcode Instruments. [Electronic resource]. – Access mode: <https://www.avanderlee.com/debugging/xcode-instruments-time-profiler/>.
24. ML Kit FrameWork. [Electronic resource]. – Access mode: <https://stfalcon.com/en/blog/post/ios-ml-kit>.
25. TensorFlow Lite: An Open Source Deep Learning Framework for Handheld Devices. [Electronic resource]. – Access mode: <https://www.opensourceforu.com/2021/04/tensorflow-lite-an-open-source-deep-learning-framework-for-handheld-devices/>.
26. AffectNet Dataset. [Electronic resource]. – Access mode: <http://mohammadmahoor.com/affectnet/>.
27. EMOTIC Dataset. [Electronic resource]. – Access mode: <https://github.com/rkosti/emotic>.
28. FER2013. [Electronic resource]. – Access mode: <https://www.kaggle.com/datasets/msambare/fer2013/>.
29. Emotion Classification. [Electronic resource]. – Access mode: <https://medium.com/@manyakakkar1920/emotion-recognitionemotion-recognition-using-machine-learning-2c7ac0dfcf3>.
30. Simple and Straightforward Technique to View FER2013 Dataset Statistics. [Electronic resource]. – Access mode: https://medium.com/@setiawanhadi_24058/simple-technique-to-view-fer2013-dataset-statistics-2b33075934c5.

ДОДАТОК А
Шари CNN на базі архітектури VGG

Layer (type)	Output Shape	Param
rescaling	(None, 48, 48, 1)	0
conv2d	(None, 46, 46, 64)	640
batch_normalization	(None, 46, 46, 64)	256
conv2d_1	(None, 46, 46, 64)	36,928
batch_normalization_1	(None, 46, 46, 64)	256
max_pooling2d	(None, 23, 23, 64)	0
dropout	(None, 23, 23, 64)	0
conv2d_2	(None, 23, 23, 128)	73,856
batch_normalization_2	(None, 23, 23, 128)	512
conv2d_3	(None, 23, 23, 128)	147,584
batch_normalization_3	(None, 23, 23, 128)	512
max_pooling2d_1	(None, 11, 11, 128)	0
dropout_1	(None, 11, 11, 128)	0
conv2d_4	(None, 11, 11, 256)	295,168
batch_normalization_4	(None, 11, 11, 256)	1,024
conv2d_5	(None, 11, 11, 256)	590,080
batch_normalization_5	(None, 11, 11, 256)	1,024
max_pooling2d_2	(None, 5, 5, 256)	0
dropout_2 (Dropout)	(None, 5, 5, 256)	0
conv2d_6 (Conv2D)	(None, 5, 5, 512)	1,180,160
batch_normalization_6	(None, 5, 5, 512)	2,048
conv2d_7 (Conv2D)	(None, 5, 5, 512)	2,359,808
batch_normalization_7	(None, 5, 5, 512)	2,048
max_pooling2d_3	(None, 2, 2, 512)	0
dropout_3 (Dropout)	(None, 2, 2, 512)	0
dropout_3 (Dropout)	(None, 2, 2, 512)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 1024)	2,098,176
dropout_4 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 256)	262,400
dense_2 (Dense)	(None, 7)	1,799

ДОДАТОК Б
Фрагменти текстів програми

Б1. Фрагмент коду програми по з творенню та тренуванню моделі ResNet-50

```
# Налаштування генератора зображень для тренування
train_datagen = ImageDataGenerator(
    rescale=1./255, # Нормалізація пікселів до діапазону [0, 1]
    validation_split=0.2, # Відокремлення 20% даних для валідації
    rotation_range=5, # Випадкове обертання зображень на 5 градусів
    width_shift_range=0.2, # Зміщення по ширині
    height_shift_range=0.2, # Зміщення по висоті
    shear_range=0.2, # Зсув зображень
    horizontal_flip=True, # Горизонтальне віддзеркалення
    vertical_flip=True, # Вертикальне віддзеркалення
    fill_mode='nearest' # Заповнення прогалин пікселями, схожими на найближчі
)

# Завантаження моделі ResNet50 без верхніх шарів
base_model = tf.keras.applications.ResNet50(
    include_top=False, # Без повнозв'язного шару
    weights=None, # Без попередньо натренованих ваг
    input_shape=(48, 48, 3) # Вхідний розмір зображення
)

# Завантаження попередньо натренованих ваг з набору датасетів Kaggle
weights_path = '../input/resnet50-weights/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5'
base_model.load_weights(weights_path)

# Заморожування всіх шарів, крім останніх 4
for layer in base_model.layers[:-4]:
    layer.trainable = False

# Побудова нової моделі на основі ResNet50
model = Sequential()
model.add(base_model) # Додавання базової моделі
model.add(Dropout(0.5)) # Регуляризація Dropout
model.add(Flatten()) # Перетворення у вектор
model.add(BatchNormalization()) # Нормалізація пакетів
model.add(Dense(32, kernel_initializer='he_uniform')) # Повнозв'язний шар з 32 нейронами
model.add(BatchNormalization())
model.add(Activation('relu')) # Активація ReLU
model.add(Dropout(0.5))
model.add(Dense(32, kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
```

```

model.add(Dense(32, kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dense(7, activation='softmax')) # Вихідний шар з 7 класами

model.summary() # Виведення архітектури моделі у консоль

# Метрики моделі
METRICS = [
    tf.keras.metrics.BinaryAccuracy(name='accuracy'), # Точність
    tf.keras.metrics.Precision(name='precision'), # Прецизійність
    tf.keras.metrics.Recall(name='recall'), # Повнота
    tf.keras.metrics.AUC(name='auc'), # Площа під кривою
    f1_score # F1-міра
]

# Callback для динамічного зменшення швидкості навчання
lrd = ReduceLROnPlateau(monitor='val_loss', patience=20, verbose=1, factor=0.50,
min_lr=1e-10)

# Callback для збереження найкращої моделі
mcp = ModelCheckpoint('model.h5')

# Callback для ранньої зупинки навчання
es = EarlyStopping(verbose=1, patience=20)

# Callback для збереження найкращої моделі за мінімальними втратами
checkpoint = ModelCheckpoint('/kaggle/working/best_model.h5', monitor='val_loss',
save_best_only=True, mode='min')

# Компіляція моделі
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=METRICS)

# Навчання моделі на 50 епохах
history = model.fit(
    train_dataset,
    validation_data=valid_dataset,
    epochs=50,
    verbose=1,
    callbacks=[lrd, mcp, es, checkpoint]
)

# Функція для побудови графіків тренування
def Train_Val_Plot(acc, val_acc, loss, val_loss, auc, val_auc, precision,
val_precision, f1, val_f1):
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))

    # Графік точності
    ax1.plot(range(1, len(acc) + 1), acc)
    ax1.plot(range(1, len(val_acc) + 1), val_acc)
    ax1.set_title('Графік отриманої точності (Accuracy)')

```

```

ax1.set_xlabel('Епохи')
ax1.set_ylabel('Точність')
ax1.legend(['training', 'validation'])

# Графік втрат
ax2.plot(range(1, len(loss) + 1), loss)
ax2.plot(range(1, len(val_loss) + 1), val_loss)
ax2.set_title('Графік показників втрат')
ax2.set_xlabel('Епохи')
ax2.set_ylabel('Втрати')
ax2.legend(['training', 'validation'])

plt.show()

# Виведення ключів історії тренування
print(history.history.keys())

# Побудова графіків тренування
Train_Val_Plot(
    history.history['accuracy'], history.history['val_accuracy'],
    history.history['loss'], history.history['val_loss'],
    history.history['auc'], history.history['val_auc'],
    history.history['precision'], history.history['val_precision'],
    history.history['f1_score'], history.history['val_f1_score']
)

```

Б2. Фрагменти кода програми по з творенню та тренуванню оптимізованої моделі на базі VGG-16

```

# Файл з даними
data_file = "../input/fer2013-csv/fer2013.csv"

# Словник для емоцій
emotions = {
    0: 'Angry', # Злість
    1: 'Disgust', # Огида
    2: 'Fear', # Страх
    3: 'Happy', # Щастя
    4: 'Sad', # Смуток
    5: 'Surprise', # Подив
    6: 'Neutral' # Нейтральний
}
num_classes = len(emotions)

# Завантажуємо дані
images, labels = load_data(data_file)

# Функція для візуалізації розподілу зображень по класах

```

```

def plot_sample_distribution(labels):
    classes, cnts = np.unique(labels, return_counts=True)
    plt.figure(figsize=(12, 5))
    plt.barh(list(emotions.values()), cnts, height=0.6)
    for i, v in enumerate(cnts):
        plt.text(v, i, ' ' + str(v), va='center')
    plt.xlabel('Кількість')
    plt.title("Розподіл зображень по класам")
    plt.show()

# Виводимо розподіл класів
plot_sample_distribution(labels)

# Поділяємо дані на тренувальний та тестовий набори
image_train, image_test, label_train, label_test = train_test_split(images,
labels, test_size=0.2, random_state=42)
image_train, image_val, label_train, label_val = train_test_split(image_train,
label_train, test_size=0.2, random_state=42)

# Виводимо кількість зразків у тренувальному, валідаційному та тестовому наборах
print('Training samples:', label_train.shape[0])
print('Validation samples:', label_val.shape[0])
print('Test samples:', label_test.shape[0])

# Функція для збільшення вибірки з використанням SMOTE
def upsampling(x, y, strategy):
    (n, w, h) = x.shape
    sm = SMOTE(sampling_strategy=strategy, random_state=42)
    x_flat = x.reshape((n, -1))
    x_up, y_up = sm.fit_resample(x_flat, y)
    n_up = x_up.shape[0]
    x_up = x_up.reshape((n_up, w, h))
    return x_up, y_up

# Збільшуємо тренувальну вибірку
image_train_up, label_train_up = upsampling(image_train, label_train, 'auto')

# Функція для one-hot кодування міток
def one_hot_encoding(labels, num_classes):
    return tf.keras.utils.to_categorical(labels, num_classes)

# Функція для зміни розміру зображень і додавання каналу
def reshape_images(images, channel=1, resize=None):
    x = tf.expand_dims(tf.convert_to_tensor(images), axis=3)
    if channel > 1:
        x = tf.repeat(x, channel, axis=3)
    if resize is not None:
        x = tf.image.resize(x, resize)
    return x

# Функція для попередньої обробки зображень та міток

```

```

def pre_processing(images, labels, num_classes, channel=1, resize=None,
one_hot=True):
    x = reshape_images(images, channel, resize)
    y = one_hot_encoding(labels, num_classes)
    return x, y

# Функція для створення callback для збереження моделі
def model_checkpoint_cb(file_path):
    return ModelCheckpoint(
        file_path, monitor='val_accuracy', mode='max',
        save_best_only=True, save_weights_only=True
    )

# Обробляємо тренувальні, валідаційні та тестові дані
x_train, y_train = pre_processing(image_train_up, label_train_up, num_classes)
x_val, y_val = pre_processing(image_val, label_val, num_classes)
x_test, y_test = pre_processing(image_test, label_test, num_classes)

# Генератори для тренування та валідації
train_datagen = ImageDataGenerator(
    rotation_range=30,
    shear_range=0.2,
    width_shift_range=0.2,
    height_shift_range=0.2,
    zoom_range=0.1,
    horizontal_flip=True
)
val_datagen = ImageDataGenerator() # Без змін для валідаційних даних

# Параметри тренування
batch_size = 128
train_generator = train_datagen.flow(x_train, y_train, batch_size=batch_size)
val_generator = val_datagen.flow(x_val, y_val)

steps_per_epoch = train_generator.n / train_generator.batch_size # Кількість
кроків на етап тренування
input_shape = x_train[0].shape # Розмір вхідних зображень

# Створення моделі
model = Sequential([ # Послідовна модель
    Rescaling(1./255, input_shape=input_shape), # Масштабування зображень
    Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal'), #
    Конволюційний шар
    BatchNormalization(), # Нормалізація
    Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same'), # Конволюційний шар
    BatchNormalization(), # Нормалізація
    MaxPool2D(), # Пулінг
    Dropout(0.5), # Випадкове вимикання 50% нейронів
    # Інші шари

```

```

    Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same'),
    BatchNormalization(),
    Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same'),
    BatchNormalization(),
    MaxPool2D(),
    Dropout(0.4),
    Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same'),
    BatchNormalization(),
    Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same'),
    BatchNormalization(),
    MaxPool2D(),
    Dropout(0.5),
    Conv2D(512, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same'),
    BatchNormalization(),
    Conv2D(512, (3, 3), activation='relu', kernel_initializer='he_normal',
padding='same'),
    BatchNormalization(),
    MaxPool2D(),
    Dropout(0.4),
    Flatten(), # Розгладжування
    Dense(1024, activation='relu'), # Повнозв'язний шар
    Dropout(0.5), # Випадкове вимикання
    Dense(256, activation='relu'), # Повнозв'язний шар
    Dense(num_classes, activation='softmax') # Вихідний шар з softmax
])

# Компілюємо модель
model.compile(
    optimizer=Adam(learning_rate=1e-3),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Параметри тренування
epochs = 20 # Кількість епох
cp = ModelCheckpoint(
    'vggnet_checkpoint.weights.h5', # Шлях для збереження ваг
    monitor='val_accuracy',
    mode='max',
    save_best_only=True,
    save_weights_only=True)

lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, verbose=1,
min_lr=1e-10) # Зменшення швидкості навчання

es = EarlyStopping(monitor='val_loss', verbose=1, patience=20)

```

```

checkpoint = ModelCheckpoint('/kaggle/working/best_vggnet_model.keras',
monitor='val_loss', save_best_only=True, mode='min')

steps_per_epoch = int(steps_per_epoch)

# Тренування моделі
history = model.fit(
    train_generator,
    steps_per_epoch=steps_per_epoch,
    epochs=epochs,
    validation_data=val_generator,
    callbacks=[lr, es, cp, checkpoint]
)

# Функція для візуалізації метрик
def plot_metrics(history):
    metrics = ['loss', 'accuracy']
    plt.figure(figsize=(15, 6))
    for n, metric in enumerate(metrics):
        plt.subplot(1, 2, n + 1)
        plt.plot(history.epoch, history.history[metric], label='Training', lw=3,
color='navy')
        plt.plot(history.epoch, history.history['val_' + metric], lw=3,
label='Validation', color='deeppink')
        plt.xlabel('Epoch')
        plt.ylabel(metric.capitalize())
        plt.legend()
    plt.show()

# Виводимо графіки метрик
plot_metrics(history)

# Отримуємо передбачення моделі на тестовому наборі
y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1) # Отримуємо класи з найвищою
ймовірністю

# Створюємо матрицю невідповідностей
cm = confusion_matrix(np.argmax(y_test, axis=1), y_pred_classes)

# Нормалізуємо матрицю невідповідностей cm_normalized = cm.astype('float') /
cm.sum(axis=1)[:, np.newaxis]

class_labels = list(emotions.values())

# Візуалізуємо матрицю похибок
plt.figure(figsize=(10, 7))
sns.heatmap(cm_normalized, annot=True, fmt='.2f', cmap='Blues',
xticklabels=class_labels, yticklabels=class_labels)
plt.title('Матриця похибок') # Заголовок

```

```

plt.xlabel('Предбачення') # Опис осі X
plt.ylabel('Наявні мітки') # Опис осі Y
plt.show()

# Імпортуємо бібліотеки для обчислення метрик
from sklearn.metrics import precision_score, recall_score, f1_score,
accuracy_score

# Отримуємо метки класів для тестового набору
y_true_classes = np.argmax(y_test, axis=1) # Перетворюємо one-hot метки в
індекси класів

# Отримуємо передбачені класи
y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)

# Обчислюємо метрики для кожного класу
precision = precision_score(y_true_classes, y_pred_classes, average=None)
recall = recall_score(y_true_classes, y_pred_classes, average=None)
f1 = f1_score(y_true_classes, y_pred_classes, average=None)

# Виводимо метрики для кожного класу
print("Метрики для кожного класу:")
for i, (p, r, f) in enumerate(zip(precision, recall, f1)):
    print(f"Клас {i}: Precision={p:.2f}, Recall={r:.2f}, F1={f:.2f}")

# Виводимо середні метрики
print("\nСредние метрики:")
print(f"Средняя точность (macro): {precision_score(y_true_classes,
y_pred_classes, average='macro'):.2f}")
print(f"Средняя полнота (macro): {recall_score(y_true_classes, y_pred_classes,
average='macro'):.2f}")
print(f"Средняя F1-мера (macro): {f1_score(y_true_classes, y_pred_classes,
average='macro'):.2f}")

# Визначаємо точність розпізнавання
accuracy = accuracy_score(y_true_classes, y_pred_classes)
print(f"\nТочність распознавания (Accuracy): {accuracy:.2f}")

```

ДОДАТОК В
Слайди презентації



Рисунок В.1 – Слайд 1

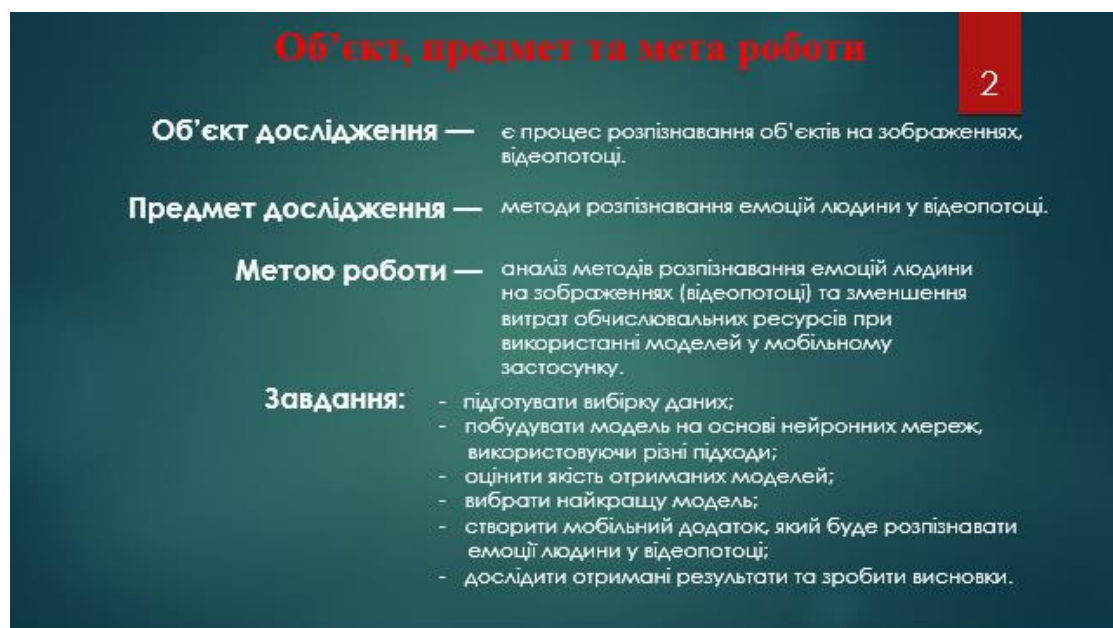
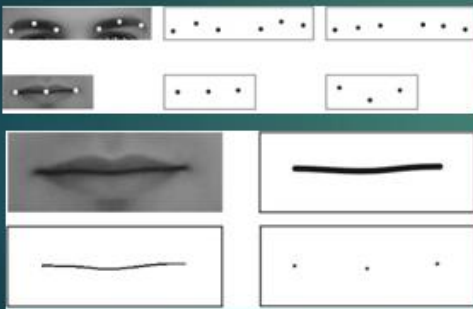


Рисунок В.2 – Слайд 2

Досліджування процесу розпізнавання емоцій на основі методу ключових точок

Процес виділення ключових точок



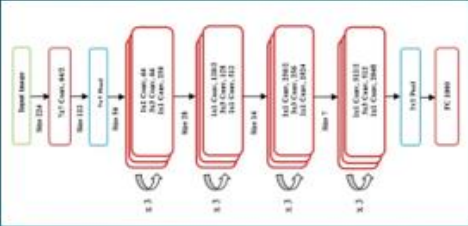
Таблиця - Класифікація емоцій за допомогою ключових точок

Емоція	Брови	Рот
Підив	Піднімаються	Відкривається
Страх	Піднімаються і зморщуються	Відкривається і розтягується
Віраза	Знижуються	Піднімається і кінці звужуються
Гнів	Знижуються і зморщуються	Відкривається або кінці знижуються
Щастя	Піднімаються	Кінці піднімаються
Смуток	Кінці знижуються	Кінці знижуються

Рисунок В.3 – Слайд 3

Досліджування методів розпізнавання емоцій за допомогою матричних нейронних мереж

Схема розташування шарів в ResNet-50



ResNet-50 — це архітектура CNN, яка належить до сімейства ResNet (Residual Networks), серії моделей, призначених для вирішення проблем, пов'язаних із навчанням глибоких нейронних мереж (деградація глибоких нейронних мереж. Коли мережі стають глибшими, їх точність насичується, а потім швидко погіршується).

Використаємо переваги цієї мережі в для вирішення поставленої задачі за допомогою техніки трансферного навчання (Transfer Learning).

VGG — це 16-рівнева архітектура CNN, базується на понятті набагато глибшої мережі з меншими фільтрами. Вона використовує згортки 3×3 на всьому шляху, що є найменшим розміром фільтра соду, який дивиться лише на деякі сусідні пікселі. Він використовує маленькі фільтри через меншу кількість параметрів, що дозволяє додавати більше шарів, що підвищує точність розпізнавання.

Приклад архітектури нейронної мережі VGG

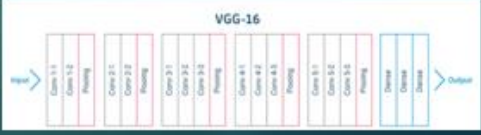


Рисунок В.4 – Слайд 4

Оптимізована архітектура мережі VGG

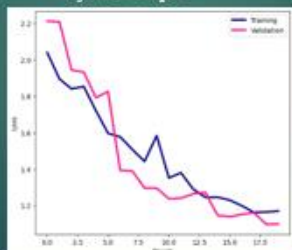
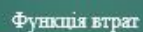
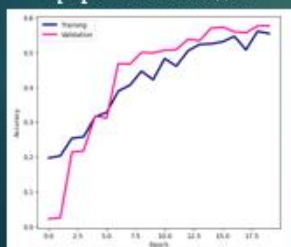
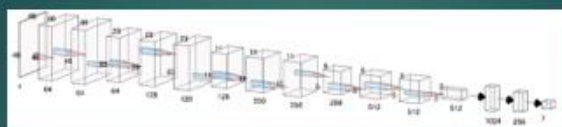


Рисунок В.5 – Слайд 5

6

Таблиця порівняння отриманих показників досліджуваних моделей

Модель	Точність, %	Повнота, %	Серед час рези́зи́в, с	Кількість параметрів	Розмір моделі (мб)
ResNet-50	70-75	68-73	0.4282	23 885 383	101,69
VGG-16	60-70	69-75	0,3833	14 628 679	58,51
Оптимізована VGG	65-75	58-64	0.2729	7 054 279	26,91

Рисунок В.6 – Слайд 6

Порівняння мов програмування та інструментів розробки				
Мова програмування	Середовище розробки	Ліцензія	Підтримка роботи з ML	Підтримка роботи з камерою
Swift	Xcode	Відкрита	Повна підтримка через Core ML	Повна підтримка
Objective-C	Xcode	Власна (Apple Proprietary)	Обмежена підтримка через Core ML	Повна підтримка
Python	Pyto, Juno	Відкрита	Обмежена підтримка через бібліотеки Core ML, TensorFlow для iOS	Обмежена підтримка через бібліотеки, такі як OpenCV
C# (Xamarin)	Xamarin Studio	Відкрита	Підтримка через ML.NET	Підтримка через Xamarin.iOS
C++	Xcode, CLion	Відкрита	Обмежена підтримка через Core ML	Підтримка через AVFoundation
JavaScript (React Native)	Visual Studio Code	Відкрита	Обмежена підтримка через бібліотеки	Підтримка через React Native Camera API
Dart (Flutter)	Visual Studio Code	Відкрита	Підтримка через сторонні бібліотеки для ML	Підтримка через Camera plugin з Flutter

Рисунок В.7 – Слайд 7

Порівняння датасетів для розпізнавання емоцій людей			
Порівняльна таблиця для 3х відомих датасетів: AffectNet, EMOTIC, FER2013			
Характеристика	AffectNet	EMOTIC	FER2013
Обсяг даних	1+ мільйон (440k анотованих)	34,000+	35,887
Кількість категорій	7 +	26 +	7
Якість зображень	Висока, реальні умови	Висока, реальні умови	Низька (48x48 пікселів)
Тип анотації	Ручна	Ручна, змішаний підхід	Ручна

Для навчання моделей було вибрано датасет «FER2013»

Рисунок В.8 – Слайд 8



Рисунок В.9 – Слайд 9

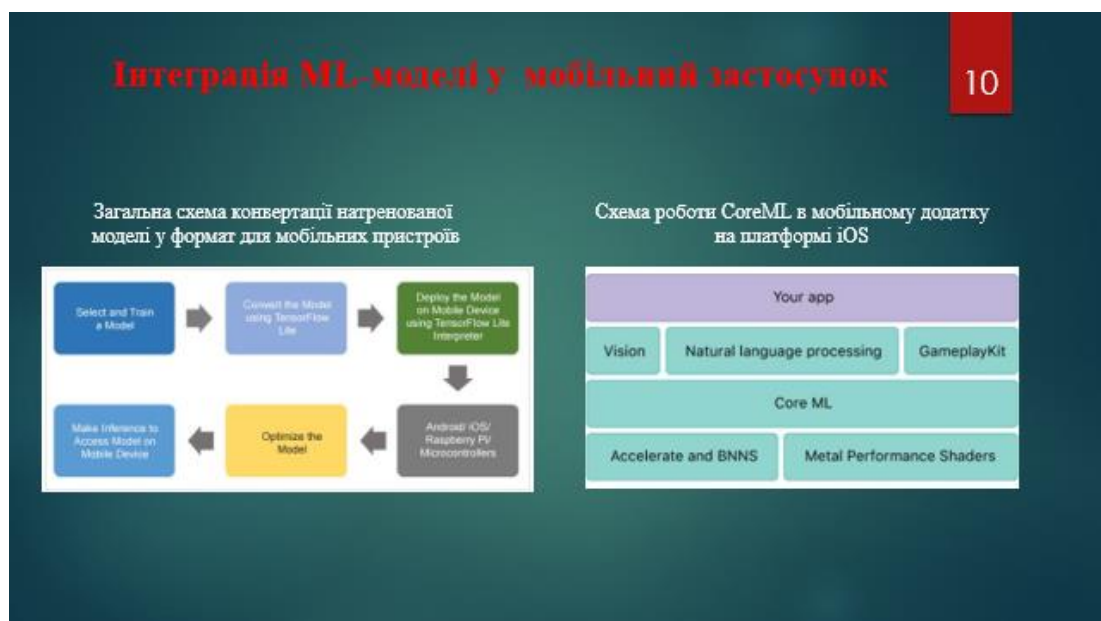


Рисунок В.10 – Слайд 10

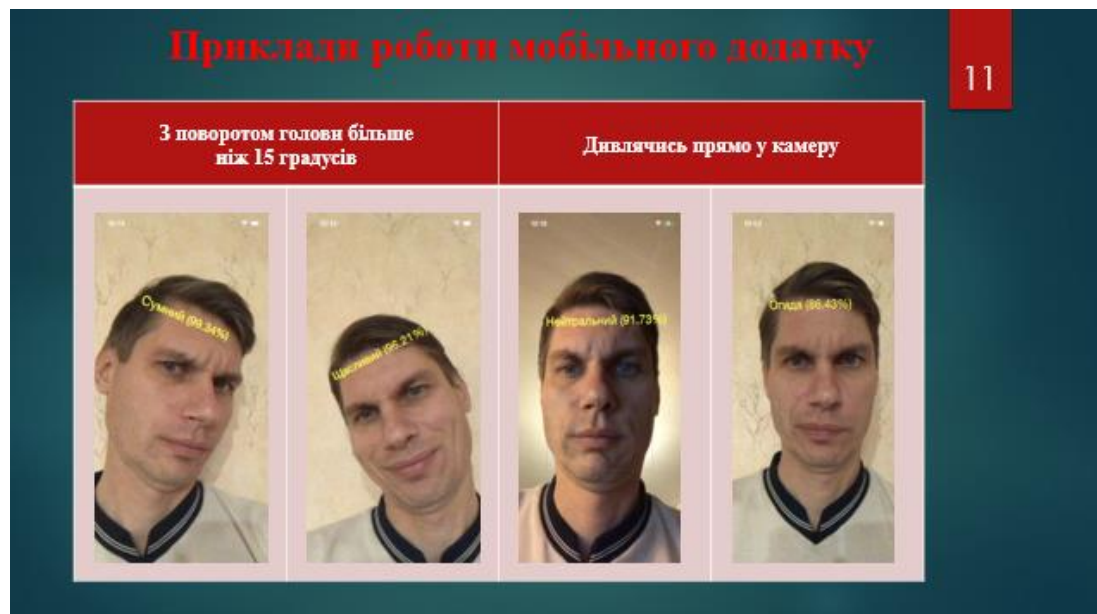


Рисунок В.11 – Слайд 11

Висновки

Результати, отримані в роботі, є рішенням практичної задачі підвищення ефективності прогнозування емоцій людини в реальному часі. Побудувавши кілька часто зустрічаються у завданнях класифікації моделей можна зробити висновок про їх ефективність в рамках роботи з розпізнавання емоцій людини за допомогою комп'ютерного зору.

Проведене дослідження продемонструвало, що завдання розпізнавання емоцій людини можна вирішити кількома сучасними методами, кожен з яких здатний показати вартий уваги результат.

Завдяки існуючим бібліотекам Apple Core ML та TensorFlow тепер стало можливим перенести нейромережі з хмари на мобільні пристрої. Незважаючи на те, що вимоги до обчислювальних ресурсів і розміру мереж, доволі високі, вони пропонують набори інструментів, які надають корисні нові можливості в програмах смартфонів. І є багато переваг у виконанні машинного навчання локально, а не в хмарі, включаючи меншу затримку, більшу доступність, покращену конфіденційність та менші витрати.

Розглянуті моделі нейронних мереж можуть бути додатково доопрацьовані та використані як модуль розпізнавання емоцій у програмних продуктах.

Рисунок В.12 – Слайд 12